DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93943-5002

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

MODERN HARDWARE TECHNOLOGIES AND SOFTWARE TECHNIQUES FOR ONLINE DATABASE STORAGE AND ACCESS

by

Christopher V. Feudo

December 1985

Thesis Advisor: David K. Hsiao

Approved for public release; distribution is unlimited

T226300

REPORT DOCUMENTATION PAGE

. REPORT SECURITY CLASSIFICATION JNCLASSIFIED	1b. RESTRICTIVE MARKINGS
. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
. DECLASSIFICATION / DOWNGRADING SCHEDULE	
PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)

. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b. OFFICE SYMBOL *(If applicable)* 52	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
. ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943-5100		7b. ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943-5100
. NAME OF FUNDING / SPONSORING ORGANIZATION	8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

. ADDRESS *(City, State, and ZIP Code)*

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

. TITLE *(Include Security Classification)*
MODERN HARDWARE TECHNOLOGIES AND SOFTWARE TECHNIQUES FOR ONLINE DATABASE STORAGE AND ACCESS

. PERSONAL AUTHOR(S)
Feudo, Christopher V.

a TYPE OF REPORT Master's Thesis	13b. TIME COVERED FROM ________ TO ________	14. DATE OF REPORT *(Year, Month, Day)* 1985 December	15 PAGE COUNT 296

SUPPLEMENTARY NOTATION

COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Magnetic, Vertical, Bubble Memory Recordings, Abstraction, Compaction, Differential Files, Data Models, Storage and Access

ABSTRACT *(Continue on reverse if necessary and identify by block number)*

omputerized data processing applications have grown over the past thirty ears to a point where they have now become a pervasive influence in our ociety. As the range of applications has grown, a continuing concern has een the cost and access time of data storage. A wide range of technologies ave been investigated to address this problem. The purpose of this thesis s to examine high-volume, on-line storage media of current and emerging echnologies and software techniques for supporting these on-line, high apacity storage media. In the first part, we analyze such media as vertical agnetic recording, thin film media, optical data disks, magneto-optic disks, ubble and Bernoulli-effect disks. Then, comparisons and evaluations of roducts and product categories are illustrated. In the second part, we eview the modern software techniques for on-line database storage and ccess.

0. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
2a. NAME OF RESPONSIBLE INDIVIDUAL rof. David K. Hsiao	22b. TELEPHONE *(Include Area Code)* (408) 646-2253	22c. OFFICE SYMBOL 52Hq

D FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

Approved for Public Release, Distribution Unlimited.

Modern Hardware Technologies and Software Techniques for On-Line Database Storage and Access

by

Christopher V. Feudo
Major, United States Army
B.S., United States Military Academy, 1972

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1985

ABSTRACT

Computerized data processing applications have grown over the past thirty years to a point where they have now become a pervasive influence in our society.

As the range of applications has grown, a continuing concern has been the cost and access time of data storage. A wide range of technologies have been investigated to address this problem.

The purpose of this thesis is to examine high-volume, on-line storage media of current and emerging technologies and software techniques for supporting these on-line, high capacity storage media. In the first part, we analyze such media as vertical magnetic recording, thin film media, optical data disks, magneto-optic disks, bubble and Bernoulli-effect disks. Then, comparisons and evaluations of products and product categories are illustrated. In the second part, we review the modern software techniques for on-line database storage and access.

TABLE OF CONTENTS

I. AN INTRODUCTION 15
  A. THE BACKGROUND 15
  B. THE ORGANIZATION OF THE THESIS 17
II. THE MAGNETIC RECORDING 19
  A. THE CONVENTIONAL RECORDING 19
    1. Operations 21
      a. The Magnetic Writing 22
      b. The Magnetic Reading 22
    2. Fixed-Head and Movable-Head Disks 23
    3. Technological Implications 29
      a. Material Requirements 31
      b. Features and Benefits of Conventional Disk 36
      c. Limitations of the Conventional Recording 39
    4. Trends and Problems in Different Taxonomies 42
III. THE BUBBLE MEMORY 46
  A. CONCEPTS 47
  B. OPERATIONS 49
  C. THE ARCHITECTURE 53
  D. BUBBLE MATERIALS 59
  E. ADVANTAGES 62

F. DISADVANTAGES 63

G. BUBBLE PRODUCTS 64

H. FUTURE TRENDS 66

I. SUMMARY 67

IV. THE VERTICAL RECORDING 71

A. CONCEPTS, OPERATIONS AND CHARACTERISTICS 72

B. THE ARCHITECTURE 72

C. THE VERTICAL MEDIUM 73

D. PROPERTIES OF THE VERTICAL RECORDING..................... 76

E. ADVANTAGES OF THE VERTICAL RECORDING 77

F. DISADVANTAGES OF THE VERTICAL RECORDING 77

G. FUTURE TRENDS 78

V. THE OPTICAL RECORDING 80

A. AN INTRODUCTION 80

B. PRINCIPLES OF OPERATIONS 82

C. THE ARCHITECTURE 82

D. APPLICATIONS OF OPTICAL RECORDING 86

E. CURRENT OPTICAL-RECORDING STATUS 87

F. MATERIAL REQUIREMENTS 94

G. FEATURES AND BENEFITS OF OPTICAL RECORDING 97

H. LIMITATIONS 97

I. FUTURE TRENDS 98

J. THE SUMMARY 100

VI. THE MAGNETO-OPTIC RECORDING 103

A. AN INTRODUCTION 103

B. BASIC OPERATIONS 104

C. MATERIAL REQUIREMENTS 106

D. FEATURES AND BENEFITS OF MOR 108

E. LIMITATIONS 109

F. FUTURE TRENDS AND POTENTIAL PROBLEMS.......................... 112

G. SUMMARY 113

VII. TWO OTHER RECORDING TECHNOLOGIES 117

A. RAM 117

B. RAM CHARACTERISTICS 120

C. SUMMARY OF RAM TECHNOLOGY 122

D. BERNOULLI CARTRIDGE 123

E. BERNOULLI EFFECT 124

F. SUMMARY 125

VIII. TECHNOLOGY COMPARISONS 130

IX. AN INTRODUCTION TO MODERN SOFTWARE 135

X. DATA ABSTRACTION TECHNIQUES 137

A. CURRENT ABSTRACTION APPROACHES FOR STATISTICAL ABSTRACTS 142

B. AN OVERVIEW OF THE ANTISAMPLING APPROACH 146

C. CURRENT ABSTRACTION METHODS FOR CONTENT ABSTRACTS 149

D. THE AUTOMATIC DATA ABSTRACTING 155

1. System Requirements 155

a. The Basic Unit of Data 156

b. Sentence-Selection Methods 156

c. Notions of Contextual Inference 157

(1) The Location Methods 157

(2) The Cue Method 160

d. Intersentence Reference 162

2. The Coherence Consideration 164

3. The System Configuration 164

a. The Sentencial Elimination 165

b. The Sentential Retention 165

c. Rules for Implementing the Functions in WCL 166

4. Data Structures for Automatic Abstracting 170

5. Conclusions 171

E. OTHER ABSTRACTING METHODS 173

XI. DATA ACCESS AND RETRIEVAL METHODS 181

A. THE USE OF HASHING FOR DATA ACCESS 183

1. The Linear Hashing 187

2. The Dynamic Hashing 190

3. The Linear Hashing with Partial Expansions 196

4. The Interpolation Hashing 200

5. The Extendible Hashing 202

6. The Coalesced Hashing 206

7. A Summary 209

B. THE EMPLOYMENT OF INDICES FOR PRECISE ACCESS 210

1. The Heap File Organization 215

2. The Indexed Sequential 216

3. The Direct Files 224

4. Primary and Secondary Keys 227

5. B-Trees and B+ Trees 228

6. Clustered Files 237

7. The Directory Hierarchy 239

C. THE STRUCTURE OF DATA FOR RETRIEVAL 242

1. The Multilist File Organization 243

    a. Answering Queries 243

    b. The Query Cost 244

    c. Updating Multilist Files 246

2. The Inverted File Organization 247

a. Answering Queries 248

b. The Query Cost 250

c. Updating Inverted Files 251

3. The Cellular Multilist File Organization 251

a. Answering Queries 252

4. Comparison of Multilist and Inverted Files 253

5. Other File Organizations for Retrieval 253

6. A Summary 259

XII. DATA COMPACTION TECHNIQUES 260

A. The Null-Suppression Technique 261

B. The Bit Mapping 262

C. The Run-Length Encoding Technique 263

D. The Half-Byte Packing 264

E. The Pattern Substitution 265

F. The Summary 266

XIII. DATA MODELS FOR DATABASES 267

A. HIERARCHIAL DATA MODEL 268

B. NETWORK DATA MODEL 270

C. THE RELATIONAL DATA MODEL 274

D. A SUMMARY 275

XIV. DIFFERENTIAL FILES 278

A. THE CONSTRUCTION OF A DIFFERENTIAL FILE 279

B. ADVANTAGES OF A DIFFERENTIAL FILE 279

1. The Reduction of Backup Costs 280

2. The Speedup of Recovery 282

3. An Increase of Data Availability 283

4. The Summary 283

XV. THE SUMMARY OF THE THESIS 285

INITIAL DISTRIBUTION LIST 295

LIST OF FIGURES

1. A Magnetic Disk Drive 24

2. Side View of a Fixed-Head Disk 25

2a. Characteristics of Fixed-Head Disk Units 27

3. Side View of a Movable-Head Disk 28

3a. Characteristics of Movable-Head Disk Unit 30

4. Magnetic Recording Materials 33

5. Maximum Coercivity 37

6. Features and Characteristics of IBM Disks 38

7. Limitations of Conventional Recording 40

8. Disk Storage Trends 43

8a. Rigid Disk Trends 44

9. Bubble Creation 48

10. Components of a MBM Chip and Film 50

11a. Serial Loop Memory 54

11b. Transfer Gate Major-Minor Loop 55

11c. Block Replicator Transfer System 56

11d. Block Replicator Swap System 59

12. Comparisons of Magnetic Bubble Memory 1983 65

13. Bubble Memory - Chip Capacity vs. Year 68

13a. Bubble Memory - Price/Bit vs. Year 69

14. Concepts of the Vertical Recording 74

15. Construction of Vertical Head 75

16. The Read/Write Operation of the
Optical Disk 84

17. The Architecture of an Optical Disk 85

17a. Tracking Error Detection 88

17b. Focus Error Detection 89

18. Five Applications of the
Optical Storage 90

19. Classification of Optical
Data Storage 91

19a. Two Types of Read-Only
Optical Data Storage 92

19b. Current Write-Once
Optical Data Storage Products 94

20. Features and Benefits of
Optical Storage 99

21. Future Trends of Optical
Disks vs. Magnetic Disks 102

22. The Magneto-Optical
Recording Basic Operations 105

23. Performance Characteristics of
Magneto-Optic Media 110

24. The Magneto-Optic Status 114

24a. Applications for Magneto-Optical
Storage 115

25. The Bernoulli Pumping Effect 126

25a. Bernoulli Box Cartridge 127

25b. Features and Benefits of Bernoulli Disk Drives 128

26. Density and Data-Rate Comparison 132

26a. Access, Capacity and Seek Comparisons 133

26b. Removability and Cost of Media 133

26c. Sturdiness and Archivability 134

26d. Head-Disk Gap and Track Servo 134

27. Generalization 140

28. Aggregation 141

29. Sampling vs. Antisampling 144

30. Example of the Rule Taxonomy 148

31. Standard Abstraction Form 153

32. WCL Entries vs. Semantic Weight 161

33. Word Control List 167

33a. Semantic Attributes for WCL Entries 168

33b. Syntactic Values for WCL Entries 169

34. Contents of the Work Area 172

35. System Configuration 176

36. Hash Organization 186

37. Linear Hashing Algorithm 189

38. Linear Hashing Example 191

39. Dynamic Hashing 194

40. Linear Hashing with Partial Expansions. Expansions 198

41. Extendible Hashing 205

42. An Example of Nondense Indexing 213

43. An Example of Multi-Level Indexing 214

44. Index Sequential File Organization 219

45. An Example of Primary and
Secondary Keys 229

46. An Example of a Binary
Tree organization 232

47. An Example of a Directory 241

48. An Example of a Multilist File 245

49. An Example of an Inverted List 249

50. Comparison of Multilist and
Inverted File Organization 254

51. The Ring Structure 257

52. The Multitiring Structure 258

53. The Hierarchial Data Model 272

54. The Network Data Model 273

55. The Relational Data Model 276

56. The Differential File 281

I. AN INTRODUCTION

A. THE BACKGROUND

Computerized data processing applications have grown over the past thirty years to a point where they have now become a pervasive influence in our society. Early data processing systems used magnetic tape as the principal storage medium for large data files. The processing was batch-sequential on a job-by-job basis and the application was mainly accounting. These systems had only secondary impact on the operational aspects of the business. Those early computers were in sharp contrast to the data processing systems of today.

In modern data processing systems many different jobs can run concurrently with the very large capacity on-line storage (i.e., directly accessible without human intervention), data-base-oriented transaction processing, and application on every operational aspect of the business.

Over the past thirty years, since the first vacuum-column magnetic-tape transport in 1953 and the first movable-head disk drive in 1957, tape and disk devices in many configurations have been the principal means for storage of the large volumes of data required by this phenomenal aggrandizement of data processing systems. Magnetic drums and other device geometries have also been important system components, but to a lesser extent. Improvements in the cost, capacity, performance, and reliability of on-line storage devices fueled these growing systems and their application capability.

As the range of applications has grown, a continuing concern has been the cost and access time of data storage. A wide range of technologies have been investigated to address this challenge. As rapid as progress in storage technology has been, the need for more capacity with improved access has increased even faster. The use of storage technologies depends on three principal factors: cost per bit, access time and unit-device cost. The reduced cost per bit in all technologies derives primarily from an increase in the density of the material being used for recording. The lower cost per bit is also associated with an increase in the physical size of the basic storage unit. In low-end systems, the unit-product cost is crucial.

While the conventional recording, (i.e., the magnetic recording) is entering yet another phase of explosive growth in applications and technology in order to meet these stringent requirements, the optical disks have begun to challenge the magnetic media. There are pressures to break free of the limitations of magnetic storage where large volumes of data are involved. These pressures come from the continuing growth of conventional storage, existing requirements of large corporate and governmental databases, and the development of new applications such as storage of digitized documents where large volumes of data must be stored at low cost. Such applications often demand a cost, capacity and performance that is difficult to achieve magnetically. Optical storage is able to provide performance that is competitive with the performance of magnetic recording. In fact, emerging optical technologies are already capable of replacing magnetic disks in certain applications. However, there is no single technology that is right for all applications. Thus, data processing installations often have available a wide range of different storage technologies. The individual needs of each application must be analyzed to determine the appropriate technology to utilize.

B. THE ORGANIZATION OF THE THESIS

The purpose of this thesis is to examine high-volume, on-line storage media of current and emerging technologies and software techniques for supporting these on-line, high capacity storage media. This thesis has two major parts. In the first part, we analyze such media as vertical magnetic recording, thin film media, optical data disks, magneto-optic disks, bubble and Bernoulli-effect disks.

Then, comparisons and evaluations of products and product categories are illustrated. In the second part, we review the modern software techniques for on-line database storage and access. Thus, this thesis is organized into two parts:

Part I: Modern Hardware Technologies for On-Line Database Storage and Operation, and,

Part II: Modern Software Techniques for On-line Database Storage and Access.

On the hardware, this thesis consists of seven chapters. Chapter II is on magnetic recording. Chapter III is on bubble-memory recording and Chapter IV is on vertical recording. Chapter V is on optical recording. Chapter VI is on magneto-optic recording. Chapter VII is on two other recording technologies, random-access memory and the Bernoulli box. The final chapter, Chapter VIII, is on the technology comparisons.

On the software, this thesis consists of six chapters. Chapter X is on data abstraction. Chapter XI is on data access and retrieval methods. Chapter XII is on data compaction. Chapter XIII is on data models for storage. The final chapter, Chapter XIV, is on differential files.

The last chapter of the thesis, Chapter XV, is the conclusion for the hardware portion and the software portion of the thesis.

II. THE MAGNETIC RECORDING

The magnetic recording consists of the conventional recording, bubble memory recording, and vertical recording. The last two recording technologies are to be discussed in considerable details in the chapters followed.

A. THE CONVENTIONAL RECORDING

"Conventional recording is entering yet another phase of explosive growth both in applications and in technology." [Ref. 4]

For the past thirty years the conventional magnetic recording has, almost exclusively, fulfilled the data storage requirements of the data processing community. During that period of time significant advances in all aspects of conventional storage technology have resulted in very significant operational gains.

In this section, conventional recording as a surface-area technology is discussed. First, conventional recording's operation is examined. Secondly, both fixed-head and movable-head disks of conventional recording are investigated. Then, their technological implications are scrutinized. These implications include their storage-device capacity, which is a direct function of the areal density of recording, the surface area provided by the storage media, and the efficiency of their utilization. The higher storage densities have required improvement in conventional recording resolution, which has been achieved through reduction in, head-disk spacing and in medium thickness. Progress in reducing the head-to-surface spacing has been the key factor in increasing the linear density of disk storage. A boundary layer of air is used to provide an air bearing which in turn determines the spacing. The progress in air bearing technology (bearing design and the surface finish and material properties of head and medium) has reduced spacings to .25 microns with laboratory studies at spacings as low as .1 micron (one micron = 10 to the -4 inches).

The track density (track density * lineal density = areal density) of conventional disks is determined by the accuracy and tolerance of the head positioning mechanism and the transverse resolution of the read-write head, as long as an adequate signal-to-noise ratio is available as the track width is reduced. Over the same time period the track density has increased from 20 to almost 1100 tracks per inch. The placing of the servo information with the data and the utilizing of better head-disk assembly packaging will lead to further advances.

To date, advances in disk media has been made by going to thinner and smoother coatings to improve resolution and to reduce demagnetization. These advances are elaborated later in the chapter. Thin films are being pursued. With these films, it is easily possible to produce medium layers of less than 25 nanometers (a nanometer = one billionth of a meter).

In respect to conventional recording's efficiency, most of the improvements during this decade will continue to come from the increased areal density. The track density can be increased by reducingthe track widths and the linear density can be achieved through continuing improvements in recording resolution, as a result of the decreased medium thickness, the reduced head-gap and head-disk spacing, and the increased use of sophisticated signal processing and error-correction codes.

1. Operations

The Danish engineer Valdemar Poulsen exhibited the first magnetic recorder at the Paris Exposition of 1900. It came 23 years after Thomas Edison had built the phonograph. In Poulsen's device a steel piano wire was coiled on the spiral groove around the surface of a drum. An electromagnet made the contact with the wire and was free to slide along a rod being positioned parallel to the axis of the drum. The drum's rotation pulled down the electromagnet. When the current from a microphone passed through the electromagnet, a segment of the wire (where the contact was made) was magnetized in proportion to the current. Although Poulsen's invention created a sensation, the recorded signal was weak. It was not until the invention of the vacuum-tube amplifier in the 1920's that magnetic recording began its steady evolution. The piano wire evolved into plastic tape with a certain amount of magnetic material. In another configuration a rotating drum was coated with a magnetic medium on which signals could be recorded on numerous circular tracks. Each track had its own electromagnet. Such devices became memories for the first modern computers [Ref. 1].

a. The Magnetic Writing

The magnetic writing, the recording of data in a magnetic medium, is based on the same principle today that applied in Poulsen's device. If a current flows in a coil of wire, it produces a magnetic field. Thus, the magnetic writing occurs as follows: The electric current supplied to the head flows through a coil around a core of magnetic material. The core throws a magnetic field into the disk, thereby magnetizing the medium lying on the disk, i.e., writingthe data [Ref. 1].

b. The Magnetic Reading

The head that writes the data can also be used to read it. One way this is done is based on the principle of induction, formulated by Michael Faraday in 1831. In the principle of induction, a voltage is induced in an open circuit, such as, a loop of wire, by the presence of a changing magnetic field. In the case of a head positioned above a spinning magnetic disk of which data have been written, the magnetic fields are originated from the magnetized regions on the disk. During the time the head is over a single magnetized region the field is more or less uniform. Hence no voltage develops across the coil that is a part of the head. When a region passes under the head in which the magnetization of the medium reverses from one state to the other, there is a rapid change in the field. Hence a voltage pulse develops. In this way the digital data in storage are read as an analogue signal, which can be readily converted back into digital form [Ref. 1].

2. Fixed-Head and Movable-Head Disks

A magnetic disk is a direct access device which has read-write heads that can both read and write data on the surface areas of platter-shaped magnetic disks. As illustrated in Figure 1, access arms are used to place the read-write heads over the surface areas of the rotating disks. Magnetic disks are available in both fixed-head and movable-head form.

Fixed-head disks are not removed from a disk drive unit. Figure 2 depicts a side view of a fixed disk, which consists of six platters with 10 surfaces and 10 read/write heads per surface. Each surface is divided into concentric rings, called tracks. Normally, the outermost surfaces of the top and bottom platters are not

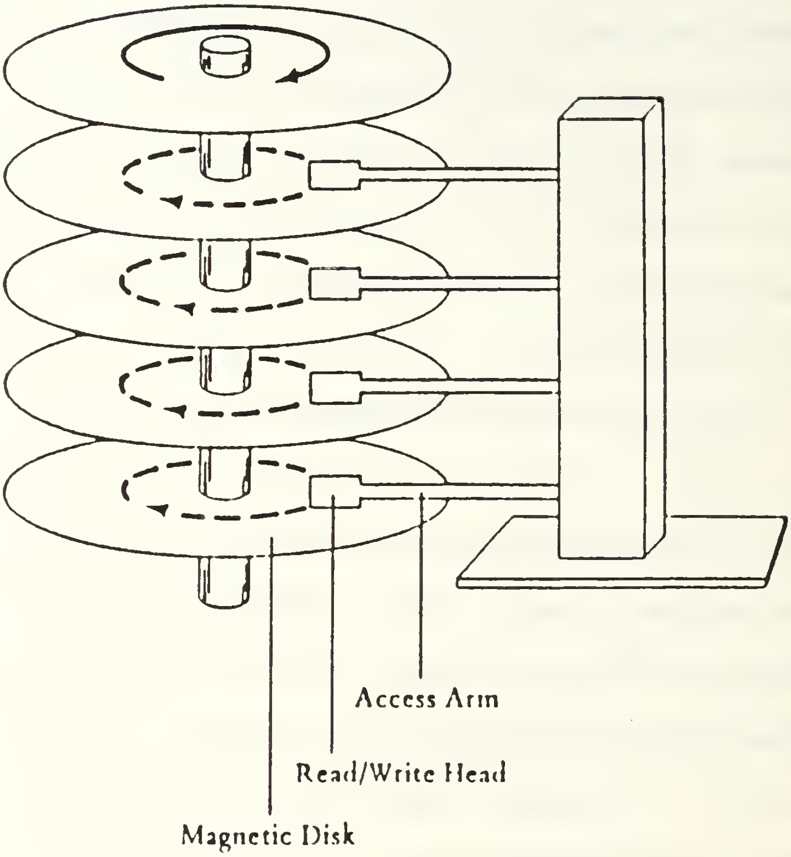

Access Arm
Read/Write Head
Magnetic Disk

Figure 1. A Magnetic Disk Drive

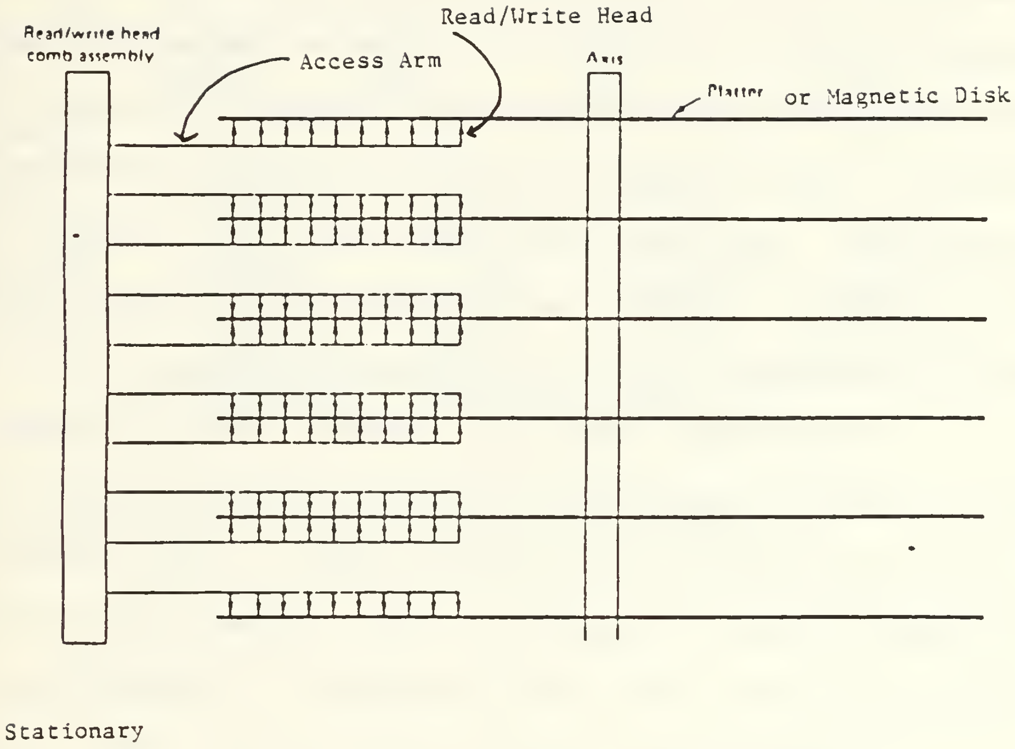
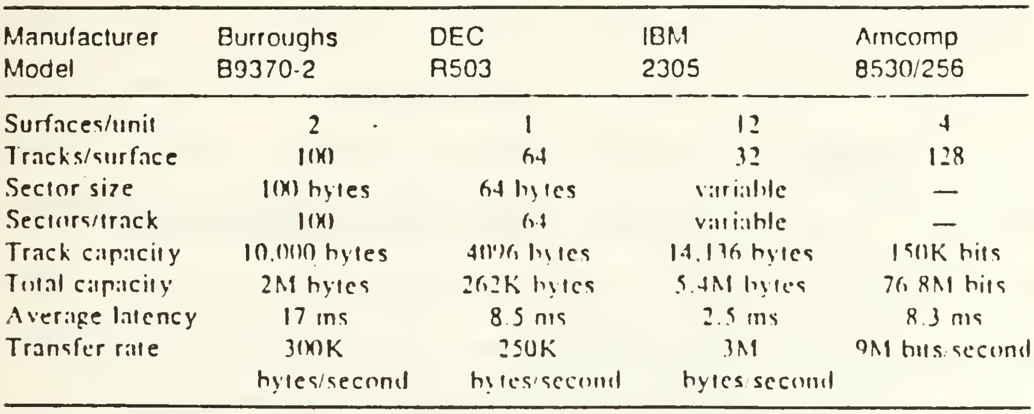

Read/Write Head
Read/write head comb assembly
Access Arm
Axis
Platter or Magnetic Disk
Stationary

Figure 2. Side View of a Fixed-Head Disk

used for storing data since they can be easily damaged. Data is recorded on the tracks by the read/write heads which are arranged on a read/write head comb assembly that is fixed in place. Since there is one read/write head per track, no seek time (i.e., the time associated with the shifting of a read/write head over a track of data) is required to move a read/write head to the proper track on a surface. This leaves only the time for rotational delay. The rotational delay is the time required to wait for the desired data to rotate under the read/write head once the read/write head is positioned over the desired track. This provides faster access time since access time is the sum of the seek time and the rotational delay.

Fixed-head disks are normally used in systems that are either dedicated to one or a few applications or when files are required to be on-line with a low access time. Characteristics of some of the commercially available fixed-head disk units are illustrated in Figure 2a.

Movable-head disks are more common than fixed-head disks because the disk packs are removable and; since there is only one read/write head per disk surface, the cost per bit of storage is less. Figure 3 depicts a side view of a movable-head disk with 10 surfaces. The read/write head comb assembly is moved in and out in order to access all of the tracks on each surface, since there is only one read/write head per disk surface. The



Manufacturer Model	Burroughs B9370-2	DEC R503	IBM 2305	Amcomp 8530/256
Surfaces/unit	2	1	12	4
Tracks/surface	100	64	32	128
Sector size	100 bytes	64 bytes	variable	—
Sectors/track	100	64	variable	—
Track capacity	10,000 bytes	4096 bytes	14,136 bytes	150K bits
Total capacity	2M bytes	262K bytes	5.4M bytes	76.8M bits
Average latency	17 ms	8.5 ms	2.5 ms	8.3 ms
Transfer rate	300K bytes/second	250K bytes/second	3M bytes/second	9M bits/second

Figure 2a. Characteristics of Fixed-Head Disk Units.

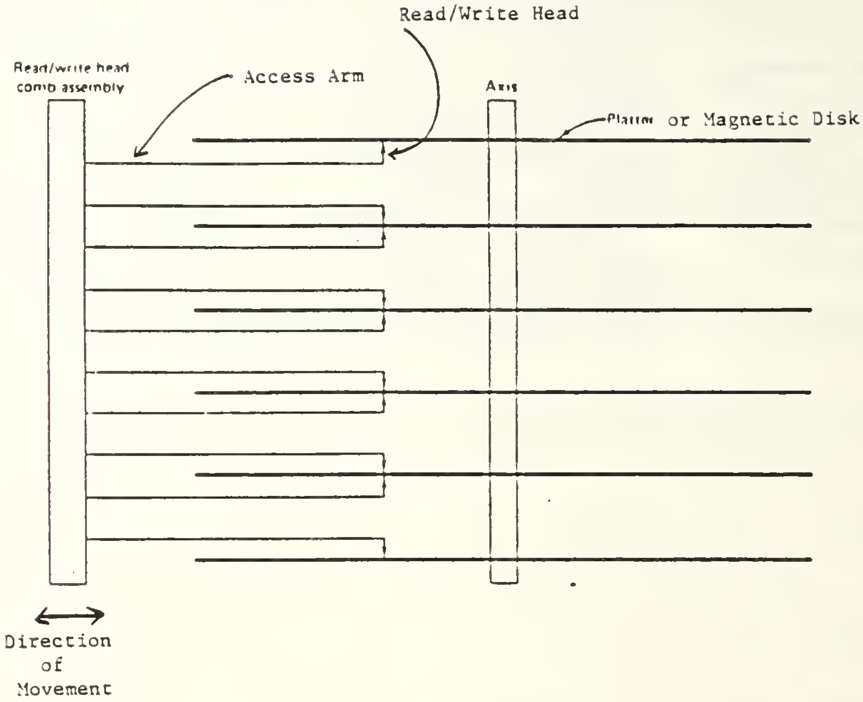

Read/Write Head
Read/write head comb assembly
Access Arm
Axis
Platter or Magnetic Disk
Direction of Movement

Figure 3. Side View of A Movable-Head Disk

read/write heads usually move together as a unit, and only one head can transfer data at a time. Thus, since the comb assembly mechanism moves, a large recording surface area can be covered with only a few read/write heads. The characteristics of some movable-head disks units are illustrated in Figure 3a.

There is yet another category of magnetic disks which is a hybrid of the above two. The Winchester-type disks, are called fixed-media direct-access storage disks. These are fixed disks, meaning that it employs a nonremovable sealed head-disk assembly, with movable-head disk units. In other words, the comb assembly, although movable, is an integral part of the disk platters. Thus, one can replace one assembly and its platters with another assembly and another set of platters. These fixed-media disks were introduced by IBM in the early 1970's with the IBM 3344, followed by the IBM 3350, and in 1979, the 3310, 3370, 3375 and in 1980 the 3380.

3. Technological Implications

Technological implications of conventional disk systems encompass the following three salient features:

(1) Material requirements,

(2) Features and benefits, and

(3) Limitations.

Manufacturer Model	HP 2100	IBM 3330	IBM 3340	CDC 33801 AZ
Surfaces/unit	4	19	12	19
Tracks/surface	200	404	696	808
Sector size	256 bytes	variable	variable	—
Sectors/track	24	variable	variable	—
Track capacity	6144 bytes	13,030 bytes	8368 bytes	13,030 bytes
Total capacity	4.9M bytes	100M bytes	69.9M bytes	400M bytes
Average latency	12.5 ms	8.3 ms	12.5 ms	16.7 ms
Transfer rate	312K bytes/second	806K bytes/second	885K bytes/second	1.2M bytes/ second

Figure 3a. Characteristics of Movable-Head Disk Unit.

a. Material Requirements

The materials for the magnetic recording medium are arranged from the top to the bottom in Figure 4 for the oldest to the newest materials available today. Of course, the writing and reading of data depend on the magnetic properties of the media in which the data is stored. The most common of these is the gamma form of the iron oxide, which is currently in use today. Iron oxide is desirable because its properties are best suited for magnetic recording and its cost is very reasonable. Moreover, its surface is uniform and homogeneous, which makes the iron oxide ideal for recording.

To use this medium in the manufacturing of a disk, the "chemical plated" process is utilized. The chemical plated process is a process by which paint-like coatings of iron-oxide particles are suspended in a polymer binder, such as the aluminum. This aluminum disk is coated with a slurry containing the iron oxide. The oxide in the slurry consists of needle-like particles approximately a micrometer (10 to the -4 centimeter) in length and a tenth of a micrometer wide. The iron atoms in each particle have their own minute magnetic fields, but the elongated shape of the particle forces the fields into an alignment along the particle's long axis. Each needle is therefore a magnetic bar, and has a dipole magnetic field. The only possible change in the field is a reversal of the north and south poles at the ends of the needle. The overall magnetization in any given region of the disk is the sum of the fields of the needlelike particles within it.

Plainly, the magnetization of a region of the disk would be maximal if its needles were aligned and if they all had their north (or their south) poles facing in the same direction. The alignment of the needles is achieved when the disk is manufactured, by rotating the disk in the presence of a magnetic field before the slurry has dried. The needles come to lie in the plane of the disk and more or less perpendicular to a radius of the disk. In an operating disk, the needles are more or less aligned with the direction of motion of the disk.

The alignment of the poles is achieved when the data is written. Specifically, it is achieved when the head applies a magnetic field to the medium. The magnetic particles are sufficiently far apart, so that their own fields do not interact appreciably with one another. However, as the strength of the applied field increases, some of the magnetic particles whose dipoles are opposite to the direction of the applied magnetic field reverse their dipole field. Ultimately, the applied field becomes strong enough to polarize all of the particles. Two complications must be noted. First, the field of the head falls off rapidly as the distance from the head increases. Second, the medium is moving, and it therefore

Fe O	(Uniform and Homogeneous)
Cr O	(Unsmooth Surface)
Cobalt-Iron Oxides	(Temp Dependent)
Barium Ferrite	(Temp Dependent)
Metals	(Unsmooth Surface)

Figure 4. Magnetic Recording Materials.

passes out of the region in which the field is strong enough to polarize the medium. It is the trailing edge of the field that governs the final orientation of the magnetization. When the field of the head is removed, the region of polarized medium remains. That is why the data can be stored. The magnetization can be driven back to zero, by reversing the flow of the current through the coil in the head and thereby applying to the magnetic medium a reversed magnetic field. Since the magnetization persists in the medium, the reversal of the magnetic field does not immediately reverse the dipoles by which the medium was magnetized in the first place until the field reaches the efficient strength.

For a magnetic medium it is desirable that the remanent magnetism (i.e., the magnetism that persists when the magnetic field is absent) be large. It also is desirable that a moderately large field strength be present to demagnetize the medium. Both of these requirements help to ensure the permanence of the stored data. In addition it is desirable that the reversal of the magnetization of this medium be accomplished over a small range of applied field strength. This helps to ensure that the states of the medium that are used for data storage will be well defined. All four of these criteria are summarized by the requirement that the hysteresis loop for the magnetic medium be large and nearly square [Ref. 1].

In addition to the iron oxide, there are four materials on the horizon as candidates for magnetic mediums, chromium oxides, cobalt-iron oxides, barium ferrites, and metals. See Figure 4. The four medium materials, although very high potential for the near future, are very limited for current usage due to their inherent disadvantages and high manufacturing cost. Disadvantages for chromium oxides include difficulty in obtaining smooth surface and good orientation. Disadvantages for cobalt-iron oxides and barium ferrites include the temperature dependence of the coercivity. Coercivity is the ability of the material to resist accidental and self-magnetization. Of course, the higher the coercivity the better the medium is for magnetic recording. Although this dependence can be reduced by varying the composition of its components. The last medium, metals, are the most promising, due to their excellent coercivity. However, metals are currently the most expensive. Metals also have other disadvantages, such as a reduction in magnetism when exposed to elevated temperatures and humidity.

Further, the four materials must be manufactured by utilizing the "sputtering" process, which is the process by which atoms or groups of atoms are ejected from a metal surface. The iron oxide does not need this process, since it utilizes the platted process. Although the sputtering process generates a very clean surface and has 3 to 8 times the capability of the platted process, it does cost approximately 60% more.

Figure 5 depicts the maximum coercivity of each of the above materials, along with the three most common anisotrophic structures of medium materials. Anisotrophy is the phenomenon of a material in which there exists preferred directions for the magnetization. Metals are not included in the figure because there does not yet exist sufficient information for comparison.

b. Features and Benefits of Conventional Disk

The features of conventional disks are illustrated in Figure 6. Benefits are listed below:

(1) lowest cost per bit as a read/write on-line storage medium,

(2) a competitive marketplace based on numerous suppliers and a large choice of product offerings,

(3) capacities up to gigabytes,

(4) read/write capability and nonvolatility,

(5) broad environmental tolerances,

(6) relatively modest entry cost,

(7) multi-billion dollar industry,

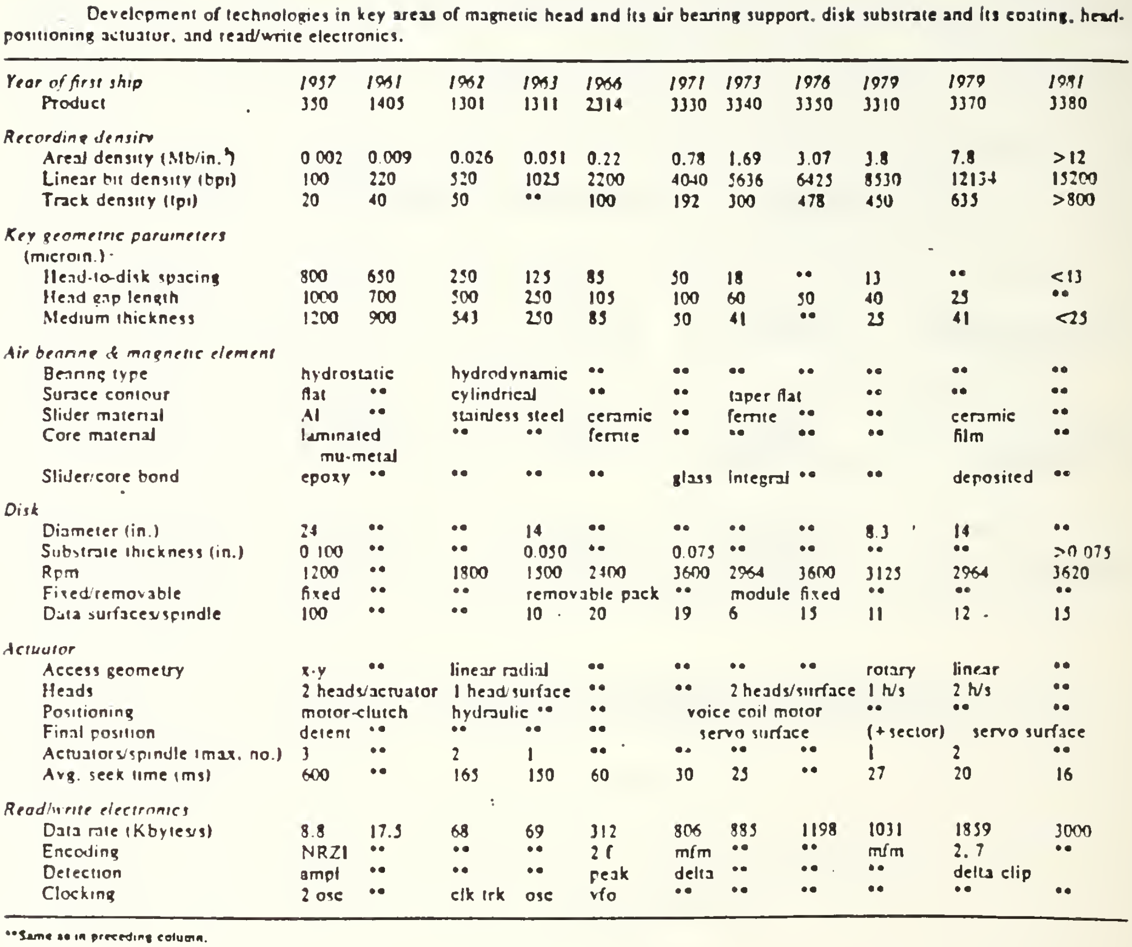
(8) established production processes,

(9) increase demand for capacity occurring faster than storage density,

(10) density still far from ultimate limits,

	Iron	Cobalt	Nickel	Fe O
Crystaline	250	3000	70	230
Strain	300	300	2000	10
Shape	5300	4400	1550	2450

Figure 5. Maximum Coercivity.



Development of technologies in key areas of magnetic head and its air bearing support, disk substrate and its coating, head-positioning actuator, and read/write electronics.

Year of first ship	*1957*	*1961*	*1962*	*1963*	*1966*	*1971*	*1973*	*1976*	*1979*	*1979*	*1981*
Product	350	1405	1301	1311	2314	3330	3340	3350	3310	3370	3380
Recording density											
Areal density (Mb/in.²)	0.002	0.009	0.026	0.051	0.22	0.78	1.69	3.07	3.8	7.8	>12
Linear bit density (bpi)	100	220	520	1025	2200	4040	5636	6425	8530	12134	15200
Track density (tpi)	20	40	50	••	100	192	300	478	450	635	>800
Key geometric parameters (microin.)											
Head-to-disk spacing	800	650	250	125	85	50	18	••	13	••	<13
Head gap length	1000	700	500	250	105	100	60	50	40	25	••
Medium thickness	1200	900	543	250	85	50	41	••	25	41	<25
Air bearing & magnetic element											
Bearing type	hydrostatic		hydrodynamic		••	••	••	••	••	••	••
Surace contour	flat	••	cylindrical		••	••	taper flat		••	••	••
Slider material	Al	••	stainless steel		ceramic	••	ferrite	••	••	ceramic	••
Core material	laminated mu-metal		••	••	ferrite	••	••	••	••	film	••
Slider/core bond	epoxy	••	••	••	••	glass	integral	••	••	deposited	••
Disk											
Diameter (in.)	24	••	••	14	••	••	••	••	8.3	14	••
Substrate thickness (in.)	0.100	••	••	0.050	••	0.075	••	••	••	••	>0.075
Rpm	1200	••	1800	1500	2400	3600	2964	3600	3125	2964	3620
Fixed/removable	fixed	••	••	removable pack		••	module	fixed	••	••	••
Data surfaces/spindle	100	••	••	10	20	19	6	15	11	12	15
Actuator											
Access geometry	x-y	••	linear radial		••	••	••	••	rotary	linear	••
Heads	2 heads/actuator		1 head/surface		••	••	2 heads/surface		1 h/s	2 h/s	••
Positioning	motor-clutch		hydraulic	••	••	voice coil motor			••	••	••
Final position	detent	••	••	••	••	servo surface			(+sector)	servo surface	
Actuators/spindle (max. no.)	3	••	2	1	••	••	••	••	1	2	••
Avg. seek time (ms)	600	••	165	150	60	30	25	••	27	20	16
Read/write electronics											
Data rate (Kbytes/s)	8.8	17.5	68	69	312	806	885	1198	1031	1859	3000
Encoding	NRZI	••	••	••	2 f	mfm	••	••	mfm	2, 7	••
Detection	ampl	••	••	••	peak	delta	••	••	••	delta clip	
Clocking	2 osc	••	clk trk	osc	vfo	••	••	••	••	••	••

••Same as in preceding column.

Figure 6. Features and Characteristics of IBM Disks

(11) a wide choice of materials, and

(12) completely reversible, inherently stable processes.

As a recording type of storage, conventional disks have the advantages of non-volatility, lower cost, direct access, allocation flexibility, a simple and reliable recording process, and allowing update in place. Their major disadvantages are two: The movable-head disks involve the mechanical motion of access assembly and long access times. the fixed-head disks incur higher production cost.

c. Limitations of the Conventional Recording

The conventional recording is limited by its physical density as depicted in Figure 7. Figure 7 shows that the current density equates to 1.2 gigabytes for the IBM 3380, with the ultimate density equating to 22.5 gigabytes. The Patty II disk system, manufactured by National Telegraph and Telephone Co., is a prototype and is to be discussed later.

The key parameters which limit the linear density are (1) the flying height of the head above the media, and (2) the physical width of the transition between neighboring, oppositely magnetized fields. The increased linear density requires a balanced reduction in these two parameters, and is ultimately limited by the failure to maintain the minimum bit-error-rate (BER) requirement for the storage device. As the linear density is increased the BER grows due to systematic

	IBM 3380	ULTIMATE	PATTY II
LINEAR DENSITY	~ 15 K bPI	40 K bPI	25.4 K bPI
TRACK DENSITY	~ 800 TPI	3K TPI	1800 TPI
AREAL DENSITY	~ 12 Mb/sq in	120 Mb/in^2	40 Mb/sq in

Figure 7. Limitations of Conventional Recording.

peak shift associated with transition crowding along the track, and/or reduced playback signal-to-noise ratio (SNR). The SNR falls due either to increased noise arising from media granularity and surface roughness, or signal loss resulting from demagnetization, combined with the need to resort to thinner layers of material in order to reduce the transition width.

At a given linear density, the track density achievable in magnetic data storage is fundamentally limited by the inductive nature of the magnetic read process. As the track width is reduced, the read signal voltage falls proportionately and the limiting track density is reached when the playback SNR falls to the critical value required to sustain acceptable BER. In practice, the achievable track density is limited by the quality of the radial positioning servo mechanism, and the degree of cross-talk due to the fringing field of the read/write head coupling to adjacent tracks. The highest track densities requires developments in all of these areas.

The linear density and track density are fundamentally linked through the SNR requirement mentioned above, with the highest track density corresponding to reduced linear density (compared to its limit) and vice-versa. The maximization of the overall areal density requires an optimum trade-off between these two parameters, depending on the media type and the magnitude of the magnetization of the storage layer, as well as the detailed performance achieved by the radial tracking servo-mechanism [Ref. 2].

4. **Trends and Problems in Different Taxonomies**

Figures 8 and 8a illustrate conventional disk trends. Note that in Figure 8, the current disk capacity of 12000 bits per inch equates to 1.3 gigabytes, and that by 1990, 5 gigabytes. This is far greater than the established doubling of storage capacity every 30 months, as been the case for the last forty years.

In Figure 8a [Ref. 3], note that IBM is experimenting with the 3380 enhanced (E), which has a storage capacity of 2.5 gigabytes per spindle, which doubles the 3380's capacity, but is far short of National Telephone and Telegraph's (NTT) Patty II capacity, which equates to 1.07 gigabytes per head disk assembly (HDA), or 8.6 gigabytes per unit, which has 8 HDA's. Moreover, the NTT driver operates at a rate greater than 40 million bits per square inch, and has a track density of 1800 tracks per inch (TPI), as well as a linear density of 25,400 bits per inch. The data rate is 4.5 megabytes per second, and the seek time is 12 milliseconds. This level of performance exceeds that of the IBM 3380 (E) in storage density by almost a factor of two and in data rate by 50%. Innovations include a sealed head disk assembly (HDA),

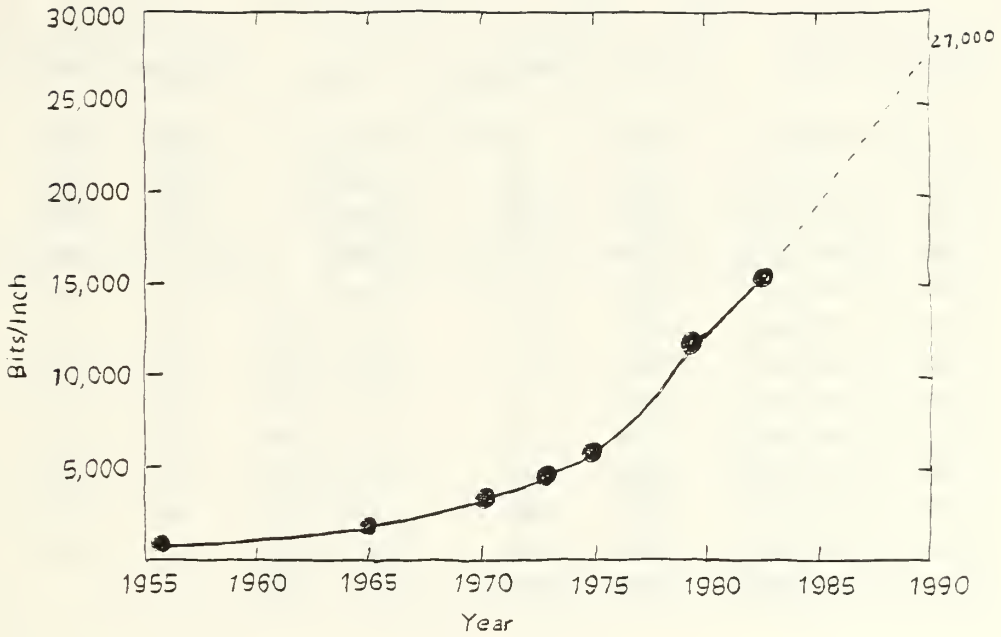


Figure 8. Disk Storage Trends

					IN MICRO INCHES		
YEAR	DEVICE	BITS/IN	TRACKS/IN	BITS/IN^2	SPACING	GAP	THICKNESS
1956	IBM 350	100	20	2000	1000	800	1200
1961	IBM 1301	500	50	25000	500	500	500
1964	IBM 2311	1100	100	110000	125	200	250
1965	IBM 2314	2200	100	220000	85	105	85
1970	IBM 3330	4040	192	775680	50	100	41
1973	IBM 3340	5600	300	1.68 x 10^6	17	60	41
1975	IBM 3350	6425	476	3.06 x 10^6	17	60	41
1978	STC 8650	6425	952	6.12 x 10^6	17	60	40
1979	IBM 3370	12134	635	7.71 x 10^6	15	24	35
1980	IBM 3380	15000	801	1.20 x 10^7	11	24	26
1981	NTT PATTY	13970	1092	1.53 x 10^7	8	32	7
1984	NTT PATTY	25400	1800	4.57 x 10^7	6	20	7
1985	IBM 3380(E)	15000+	~1400	~ 2.0+ x 10^7*			

Figure 8a. Rigid Disk Trends.

with a helium atmosphere, a thin film sputtered ferrite disk, a flying height of 0.15 microns, and a unique rotary multiactuator assembly. Both the IBM 3380 (E) and the NTT Patty II are still only prototypes.

Some of the problems that must be overcome in order to achieve the anticipated conventional recording performance are better tracking error, positioning and servoing systems developments. They are currently very poorly developed. a very high degree of accuracy is required between the position of the read/write head and the location of data on the media surface. Also, the track densities appear very unlikely to exceed 2000 TPI.

III. THE BUBBLE MEMORY

"Bubble memory, initially touted as a universal replacement for disk technology is today regarded as a technological flash in the pan." [Ref. 7]

The magnetic-bubble memory (MBM) is a solid-state magnetic memory which employs shift registers. These shift registers move magnetic domains which represent binary data. The rotating magnetic fields of the domains are used for the binary orientation of the data. Unlike conventional semi-conductor memory devices which are produced from silicon materials, MBM utilizes synthetic garnet or amorphous materials.

The ingenious technological discovery of MBM dates back to 1966 when Bell Laboratory scientists discovered that magnetic bubbles could be used to record, store, and read data by applying and manipulating external magnetic forces. The following features of the bubble phenomena aided its development as potential memory devices:

(1) Bubbles were stable over a range of the magnetic bias field (i.e., stable storage);

(2) Bubbles could be elongated by lowering the magnetic bias field for further manipulation, and

(3) Bubbles could be annihilated by raising
    the bias field.

A. CONCEPTS

Bubbles are microscopic magnetic cylinders of reverse polarization to that of the thin magnetic film substance surrounding the bubbles on a memory chip. The bubbles are the individual memory cells, only smaller, and hence, more densely packed than conventional semiconductor memory. The presence of a bubble indicates a logic digit of "1" and the absence indicates a logic digit "0".

Figure 9 depicts the technique for creating bubbles. The bubbles are created in memory chips made of two layers. The first layer is a nonmagnetic substrate of gadolinium gallium garnet about 0.015 inches thick. The second layer is an extremely thin 3 micrometer ferromagnetic single crystal of garnet grown on the substrate. The single crystal completely covers a 3 inch diameter wafer yielding up to 44, 1/4 square-inch bubble memory chips. The magnetic film crystal is magnetized at right angles to the surface so that magnetic equilibrium occurs. Wavy interspersed areas of north and south poles are created in total equal proportions. When an external magnetic field, or "bias" field as it is usually called, is imposed on the chips, magnetic regions with polarity similar to the bias field expand and those regions of

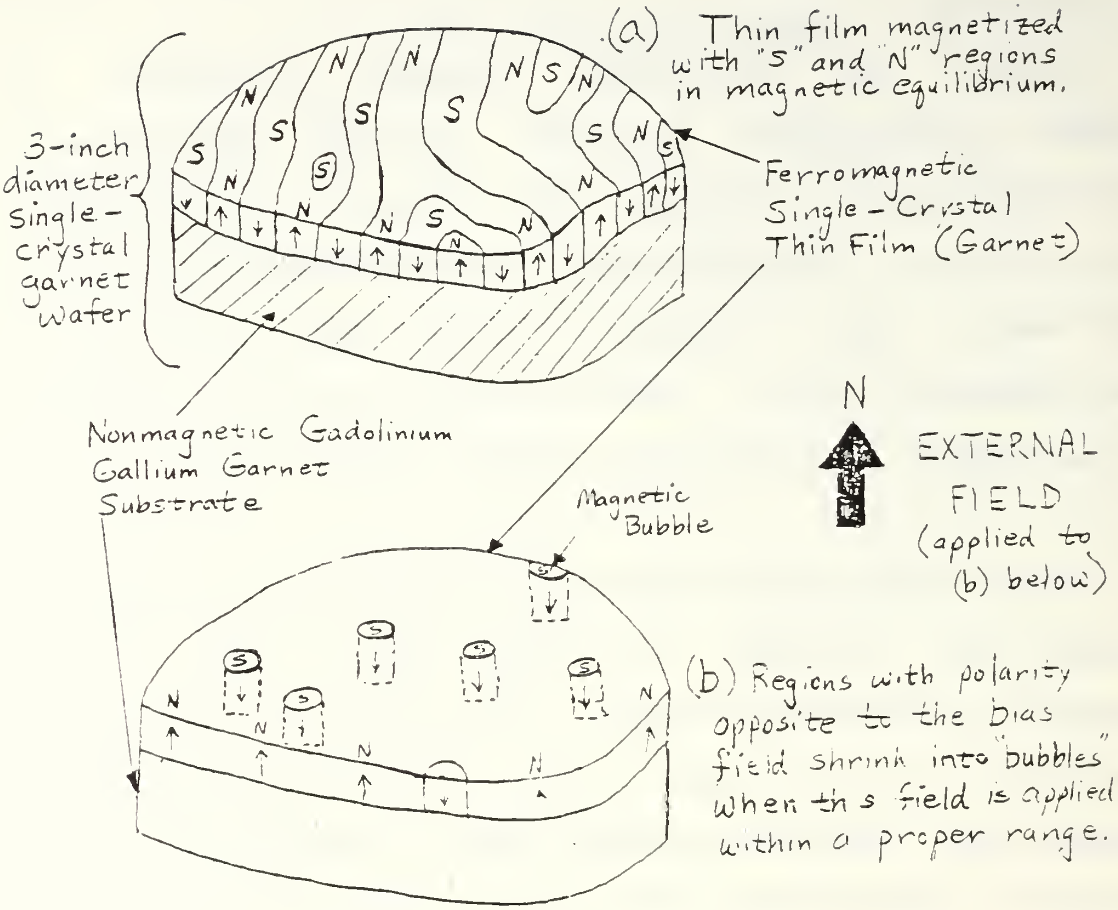

(a) Thin film magnetized with "S" and "N" regions in magnetic equilibrium.
3-inch diameter single-crystal garnet wafer
Ferromagnetic Single-Crystal Thin Film (Garnet)
Nonmagnetic Gadolinium Gallium Garnet Substrate
N
EXTERNAL FIELD (applied to (b) below)
Magnetic Bubble
(b) Regions with polarity opposite to the bias field shrink into "bubbles" when this field is applied within a proper range.
N
S

Figure 9. Bubble Creation.

reverse polarity shrink until they form tiny magnetic cylindrical bubbles. These bubbles are like small islands in an ocean of opposite magnetism. In other words, the polarity of bubbles can be either north or south poles, but are always opposite in polarity to the bias field used in the manufacturing process.

B. OPERATIONS

Figure 10 illustrates the operation of bubble memory recording. Maintaining and manipulating the bubbles around laterally throughout the film is a delicate operation. The bubbles are stable within a certain intensity range of the bias field created by two rectangular permanent magnets, one above and one below the chip to develop the perpendicular magnetic field which generates and maintains the bubbles. Above a certain range the bubbles collapse and disappear, and below this range the bubbles expand once again to form the wavy, stable , and oppositely polarized magnetic regions. A varying electromagnetic field created by a pair of electromagnetic or orthogonal coils wound around the chip at right angles to each other provides a rotating electromagnetic field that moves the bubbles laterally along a permalloy track whenever 90 degrees out-of-phase current is fed to the two coils. The permalloy tracks are laid out on the garnet film using printed-circuit techniques in chevron,

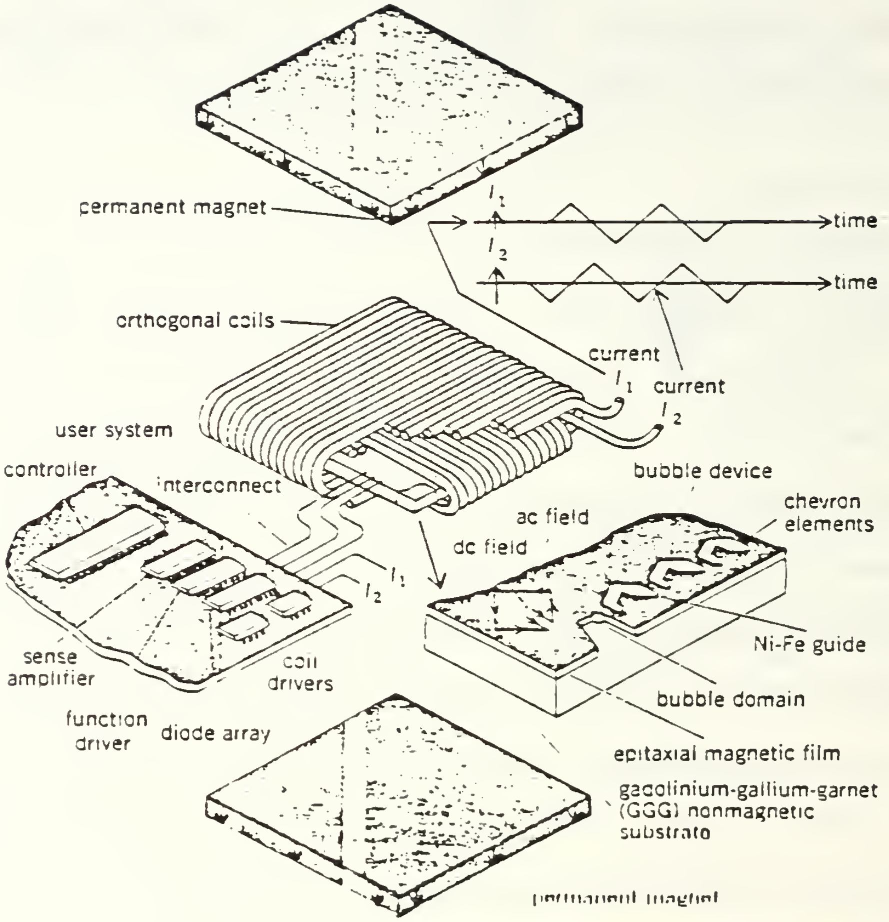

permanent magnet
I_1
time
I_2
time
orthogonal coils
current
I_1
current
I_2
user system
controller
interconnect
bubble device
chevron elements
ac field
dc field
I_1
I_2
Ni-Fe guide
sense amplifier
coil drivers
bubble domain
function driver
diode array
epitaxial magnetic film
gadolinium-gallium-garnet (GGG) nonmagnetic substrate
permanent magnet

Figure 10. Components of a MBM Chip and Film.

T-bar, or semicircle patterns. As the rotating magnetic field changes the instantaneous polarity of the track elements, the bubbles move around the track. The changes in polarity pull the bubbles through the chevron areas and down the path. A bubble moves one stage (one chevron, T-bar, or semicircle) along the track for each 360 degree revolution of the magnetic field. The bubble stream is kept in motion by passing "write" and "read" heads at different points with data being read as the bubbles make a full revolution around the track.

A MBM chip must also contain structures capable of generating, annihilating, detecting, and replicating bubbles. With such mechanisms, the basic functions for a memory can be emulated by the magnetic bubble device. This device is the controller.

Bubbles are generated by a nonmagnetic conductor loop, called the "hairpin", which is inserted between the garnet film and the special "pickax-shaped" permalloy propagation track element (chevron, T-bar, or semicircle). When a pulsed current passes through the loop, a magnetic field opposite to the bias field creates a bubble which is then passed onto the track by the effect of the rotating magnetic field. Changing bubble direction involves using the same "hairpin" and "pickax" arrangement to create field polarities which momentarily block bubble movement caused by the rotating electromagnetic field and divert it into other propagation tracks. Erasing old bubbles is accomplished similarly to the method of changing bubble direction except that instead of channeling a bubble to another storage loop track, a bubble is removed from the track, isolated, and erased by another pulse of proper polarity strong enough to to cause a bubble to collapse.

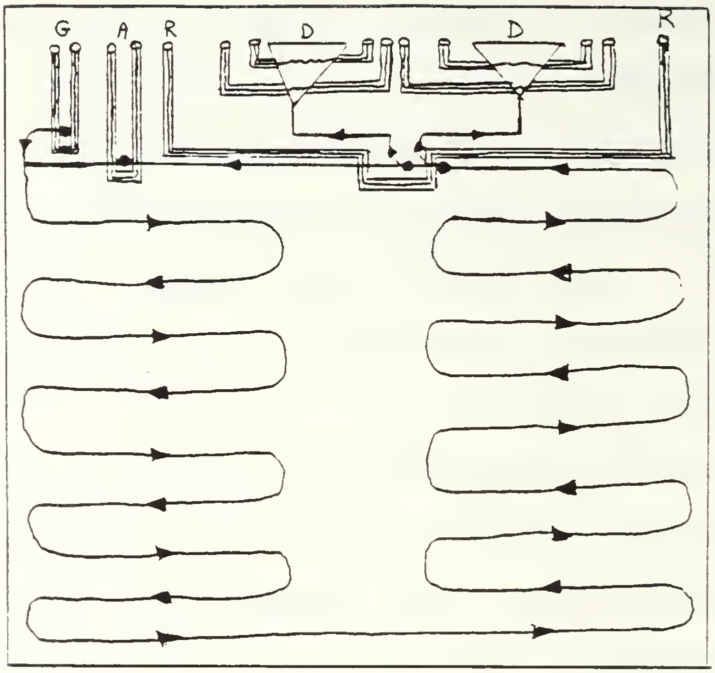
Bubble detection is either destructive or nondestructive. In destructive detection, the bubble is detected and read, but is destroyed by the read process and does not remain in memory storage. In nondestructive detection, the bubble is detected and replicated; the replication is diverted to a "read" detector where that bubble is read and then erased, and the original bubble remains in storage. In the nondestructive read process, the replicator basically splits a stretched bubble created by the "hairpin" and separates the two clones. As the rotating field operates, the two identical bubbles follow separate paths, one to remain in memory and the other to pass on to the detector and eventual erasure. The bubble passing to the detector is stretched hundreds of times in diameter and passed under magnetoresistive material. This conductive material has a resistance which varies with the strength of the surrounding magnetic field. A small milliampere current is sent through this material normally in the chip. When the bubble passes this material in the detection device, the resistance of the material drops sharply and enough current flows to produce a pulse announcing the presence of a bubble [Ref. 8].

C. THE ARCHITECTURE

There are basically two categories of architecture for MBM: the serial loop system or the major-minor loop system. The major-minor loop system has three implementation variations: transfer gate, block replicator transfer and block replicator swap.

Each of these four architectures uses function gates to generate, replicate, detect, and erase data and a pair of detectors to eliminate the effect of the rotating magnetic field. The serial loop scheme will be mentioned only briefly since it is seldom employed (see Figure 11a). The serial loop scheme consists of a single serial loop which forces the bubble stream to circulate throughout the entire loop before a bubble can be read or destroyed. Access times are typically high at around 370 ms. Detection can be destructive or nondestructive.

The first scheme of major-minor loop bubble memory architecture is the transfer gate system (see Figure 11b). The transfer gate system major-minor loop architectures are constructed with a major loop which directly connects to the generation, detection, replication, and annihilation devices for reading and writing on one side and



A- Annihilator
D- Detector
G- Generator
R- Replicator

Figure 11a. Serial Loop Memory.

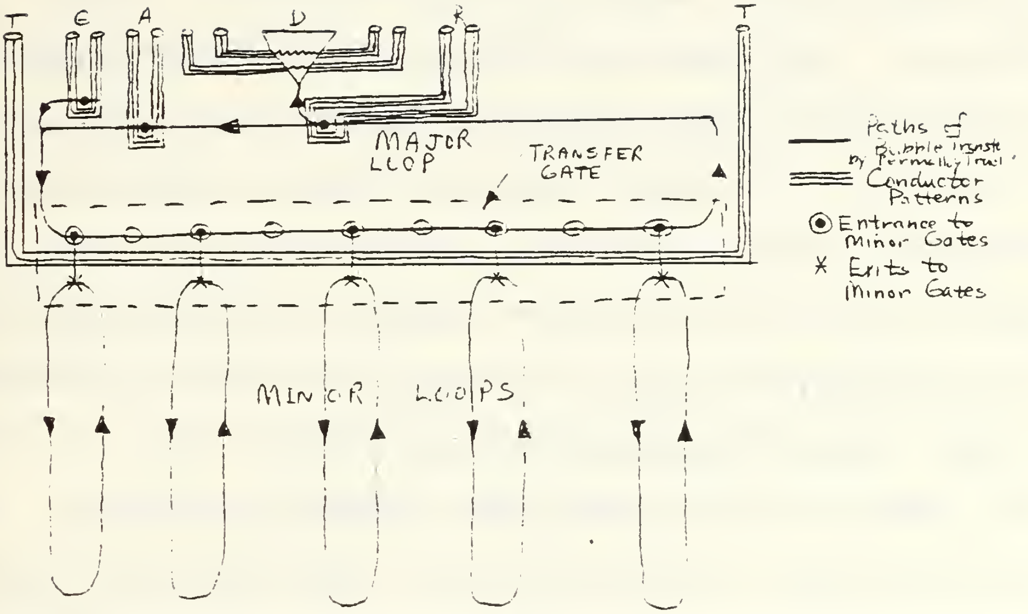

T
E
A
D
R
T
MAJOR LOOP
TRANSFER GATE
MINOR LOOPS
Paths of Bubble transfer by Permalloy Track
Conductor Patterns
Entrance to Minor Gates
Exits to Minor Gates

Figure 11b. Transfer Gate Major-Minor Loop.

to a parallel series of minor loop. This system has serial input, parallel storage, and serial output. Minor loops are connected to alternate bit positions along the major loop which are all enclosed in a transfer gate. Data is written into the major loop in alternate bits, shifted around the major loop until the first bit arrives at the first minor loop, the second data bit at the second minor loop and so on until each minor gate entrance holds a bit. Then the transfer gate is pulsed to enter data into storage. Old data must be read out serially from each minor loop at the respective minor loop exit before new data bits can occupy memory previously occupied by old data.

There is no "write-over" procedure available. Rather, old data must be destructively read before new data can be entered into the same address. Control circuitry ensures new data is inserted only into the previously vacated memory slot occupied by the old data. When data is only to be read and not replaced, it must be replicated so that one copy returns via the major loop to the minor loop storage and the other copy is read by the detector and then annihilated. As it may be surmised, the transfer gate architecture is not fast enough for some applications because of the alternate spacing between minor loops.

The second scheme of major-minor loop bubble memory architecture is the block replicator transfer system (see Figure 11c). More organizational separation of function is utilized, resulting in a scheme which is twice as fast as the transfer gate version. The major loop is divided into two write lines at one end of the minor loop bank and two output read lines at the other end. The minor loop bank is divided into even-bit and odd-bit storage banks with each bank having its own generator and major loop write line. The odd bank has an extra bubble position so that identical data bits are offset from those in the even bank. Old data is destructively read out the same basic way as the transfer gate system; however, in this architecture, auxiliary control circuitry times the rotation of minor loops and the transfer and replicate gates so that new data properly replaces old data. That is, new data is written only where old data vacant slots are located. The advantage of having the read and write functions separated is that they have their own dedicated loop connections. As soon as a vacant memory spot is available on a minor loop, new data from the write end of the minor loops can be entered into the vacancy. Consequently, there is no need to wait for outgoing data to clear the major loop before the arrival of new data. This is in direct contrast to the corresponding actions in the transfer gate scheme.

To simplify data read-out, the control circuitry collects a bubble from each minor loop at the read end of the minor loop banks where the replicator gate for each such loop is located. The block replicator is then pulsed,

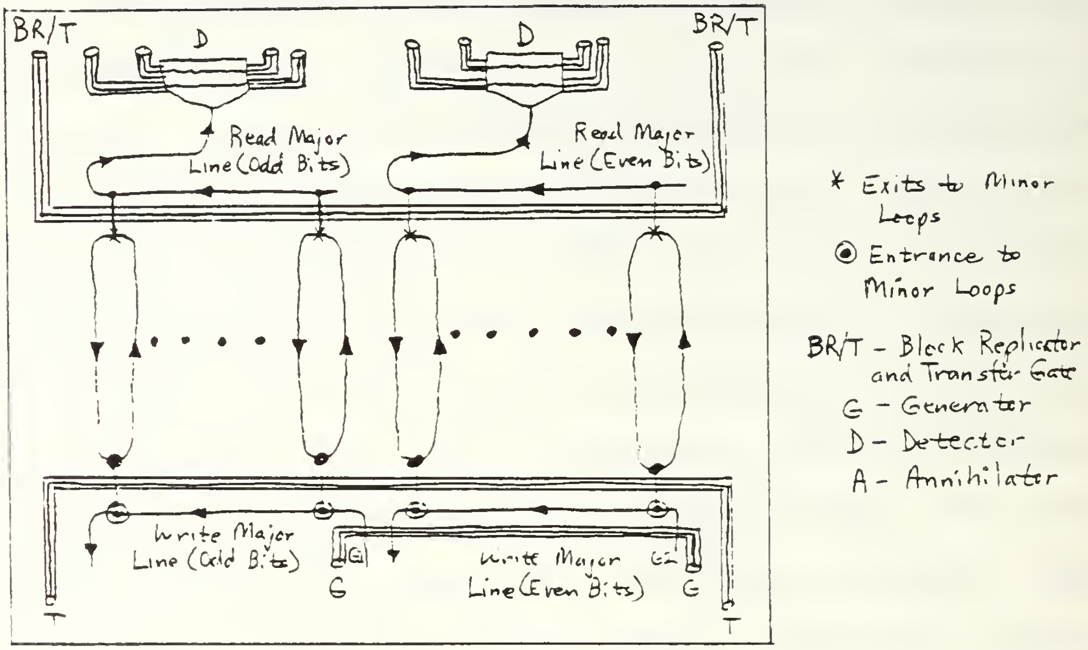

BR/T
D
D
BR/T
Read Major Line (Odd Bits)
Read Major Line (Even Bits)
Write Major Line (Odd Bits)
G1
G
Write Major Line (Even Bits)
G2
G
T
T
* Exits to Minor Loops
⊙ Entrance to Minor Loops
BR/T - Block Replicator and Transfer Gate
G - Generator
D - Detector
A - Annihilater

Figure 11c. Block Replicator Transfer System.

resulting in the replicated copies being kept in the minor loop storage. As with the transfer gate architecture, the block replicator architecture employs nondestructive detection for only reading data.

The third and final major-minor loop bubble memory architecture scheme is the swap gate scheme (see Figure 11d), which replaces the bank of transfer gates at the write end of the minor loop banks with a bank of swap gates. This bank allows old data to be transferred to write/swap exits at the same time new data is available at the swap/write entrances to the minor loops. When the swap gate is energized, new and old data merely swap places. New data is stored in the minor loops and old data is whisked away by the major write lines to be erased by the annihilator. The obvious advantage of this scheme is that a lot of data does not have to be erased before new data is written. This architecture also uses nondestructive detection for only reading data.

D. BUBBLE MATERIALS

Certain elements and their alloys (Fe, Co, Ni, Gd) along with other substances exhibit the well-known property of magnetism. This property permits a material's atoms to achieve a high degree of alignment despite the atoms' tendency towards randomization due to some type of thermal motion. Materials can be shaped such that their

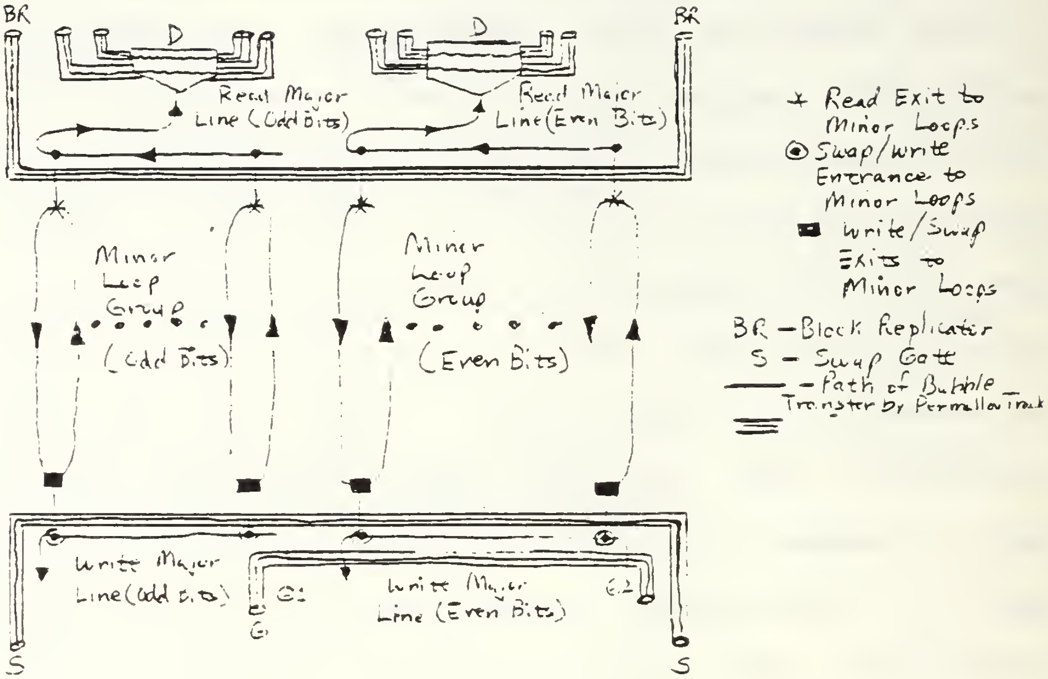

BR
D
Read Major Line (Odd Bits)
D
Read Major Line (Even Bits)
BR
Minor Loop Group (Odd Bits)
Minor Loop Group (Even Bits)
Write Major Line (Odd Bits)
G1
G
Write Major Line (Even Bits)
G2
S
S
✳ Read Exit to Minor Loops
⊙ Swap/write Entrance to Minor Loops
■ write/Swap Exits to Minor Loops
BR — Block Replicator
S — Swap Gate
— Path of Bubble Transfer by Permalloy Track

Figure 11d. Block Replicator Swap System.

direction of magnetization is along a particular direction. Several important properties of magnetism are exhibited when a magnetic substance is subjected to an external field. First, a relative increase in the external field will cause a relative increase in the substance's magnetic field. Secondly, if a single, thin, crystal film of certain magnetic materials is shaped perpendicular to the axis of the original magnetism, the results are wavy strips of matter having alternating directions of magnetization which are perpendicular to the surface of the film. Thus, it is the combination of these two properties which supplies an environment for a MBM.

Of the available bubble materials, the most common and currently most utilized is a cubic structure garnet, which include rare earth (Re), and iron (Fe). Magnetic garnet films can easily be tailored to produce specific magnetism along a desired direction, as well as to enable the coercivity to be better controlled. Also, satisfactory operation can be sustained with these garnets over a temperature range extending from room temperature up to 70-100 degrees centigrade. Moreover, the Curie temperature, which is the temperature at which demagnetization occurs, is fairly constant. This of course provides useful bubbles. The size of bubble created can vary from substances for MBM.

Other films include hexaferrites (such as BaFeO), amorphous materials (such as ReTu alloys), and orthoferrites. Hexaferrites are hexagonal, and thereby have a crystalline anistrophy, which is adequate for bubbles. They represent a class of materials with a higher coercivity than garnets and the capability of creating smaller bubbles (<.5 um). But, the structure tends to grow more rapidly perpendicular to the axis, thereby making good films more difficult. Although, its velocity is faster than garnets, its Curie temperature is more varied. Also, at this time, since only small bubbles can be created, its uses are uncertain and limited.

Amorphous films being amorphous are not single-crystals-like garnets. They are less expensive than other materials, but are too sensitive to variations in temperature. Also, its velocity is slower than the other materials. The size of bubbles is slightly better than hexaferrites (.2 to .6 um), but still much less than garnets.

The orthoferrites were the first materials to be utilized for MBM. Its magnetization is much too low to be very useful and only large bubbles can be created, in the range of 50-100 um [Ref. 11].

E. ADVANTAGES

The following are some of the advantages of bubble memory over conventional semiconductor technologies [Ref. 10]:

(1) non-volatility of memory (if there is a power loss to the bias field generating coils),

(2) high robust reliability and ability to ensure data integrity,

(3) non-mechanical aspects,

(4) ease of programmability,

(5) simpler interfaces than with disks,

(6) versatile technology in terms of architectural options,

(7) highly portable without the need of power of refresh techniques,

(8) high relative yield in major-minor loop chip manufacturing which can tolerate a degree of defect in manufacturing, and

(9) more resistant to the effects of electromagnetic pulse effects (simple magnetic shields can be used to encase devices containing bubble memory chips).

F. DISADVANTAGES

Some of the disadvantages include:

(1) high cost of technology,

(2) slower access rate (in the range of 4 to 10 ms),

(3) slow data transfer rates (in the range of 40-100 Kbits/sec),

(4) not too resistant to temperature variations (typical operating range 0 to 55 degrees in C), and

(5) non-operating storage temperature ranging only from -40 to +100 degrees in C.

G. BUBBLE PRODUCTS

Almost every major electronics company in the world was initially involved with magnetic bubble research. Either the technology was too complex or the bubble device did not hold enough business potential, many development programs were abandoned. Thus, it is no wonder that many companies dropped MBM altogether. Only Intel remains in the bubble development field.

Figure 12 compares some of the existing bubble products. The first commercially offered product was Texas Instrument's TI 92 Kbit memory module, the TIB0203, in 1978. It employed a major/minor loop architecture with 157 loops, 13 of which were redundant. TI followed this with three higher capacity units which employed a block replicate architecture. Soon Rockwell and Fujitsu also entered the market with bubble devices of their own.

Early in 1979 Intel introduced the first 1-Mbit device on the market. This device also included all the support components to turn the magnetic bubble device into a magnetic bubble system. These support elements included a controller, a formatter/sense amplifier, a coil pre-driver, a coil driver and a current pulse generator. The controller interfaces with the microprocessor system and converts microprocessor read/write commands into the necessary control signals to carry out the the selected operation within the MBM system. The formatter/sense

MANUFACTURER	DEVICE	CAPACITY	DESIGN APPROACH	SUPPORT CIRCUITS	POWER-FAIL PROTECTION	ERROR CORRECTION
TEXAS INSTRUMENTS	TIB0203	92 KBIT	COMPONENT	NONE	NONE	NO
	TIB0303	254 KBIT	COMPONENT	NONE	NONE	NO
	TIB250	256 KBIT	COMPONENT	NONE	NONE	NO
	TIB1000	1 MBIT	CONTROLLER ONLY	NONE	NONE	NO
ROCKWELL INTERNATIONAL	RBM256	256 KBIT	COMPONENT	NONE	NONE	NO
	RBM411	1 MBIT	CONTROLLER ONLY	YES	NONE	NO
NATIONAL SEMICONDUCTOR	NBM2256	256 KBIT	SYSTEM	FULL	BUILT-IN	YES
	NBM2011	1 MBIT	SYSTEM	FULL	BUILT-IN	YES
INTEL MAGNETICS	7110	1 MBIT	SYSTEM	FULL	BUILT-IN	YES
	7114	4 MBIT	SYSTEM	FULL	BUILT-IN	YES
MOTOROLA	MBM2256	256 KBIT	SYSTEM	FULL	BUILT-IN	YES
	MBM2011	1 MBIT	SYSTEM	FULL	BUILT-IN	YES
	NA	1 MBIT	SYSTEM	FULL	BUILT-IN	YES
FUJITSU	FBM32DA	64 KBIT	BUBBLE CASSETTE BOARD SET	BUBBLE CASSETTE BOARD SET	EXTERNAL POWER SEQUENCE	YES
	FBM43DA	256 KBIT	BUBBLE CASSETTE BOARD SET	BUBBLE CASSETTE BOARD SET	EXTERNAL POWER SEQUENCE	YES

Figure 12. Comparison of Magnetic Bubble Memory 1983.

amplifier has several functions. First, it contains two sense amplifiers for the detection of bubble signals produced at the detector output. Second, upon system initialization it stores the redundant loop information in an internal loop register and insures that the sense amplifiers identify the correct bits at the detector output. Third, this element contains an error correction mechanism which improves reliability of input and output. The current-pulse generator cause the control signals to enable the correct current sources for the desired operation. It also includes a power-fail circuitry to preserve the integrity of the data if power is suddenly lost. Finally, the coil driver produce the high-value currents to create the required magnetic fields. Intel was followed by National Semiconductor with a 256-Kbit device and it too had all the necessary support elements.

Today, the only US company that is still involved in this field is Intel. Intel has recently announced an enhancement of its highly sophisticated bubble memory controller (BMC). One can support up to an entire megabyte, and the other up to four megabytes [Ref. 9].

H. FUTURE TRENDS

Although MBM panacea has disintegrated, its future is not as bleak as expected. Today, military applications provide the major need for MBM. Intel has recently announced a 4-Mbit chip with the capability of storing 1 megabyte, and a 4-megabyte capability is in the near horizon. Figures 13 and 13a depict actual and anticipated trends in the chip capacity vs. the year, and the price/bit vs. the year, respectively. In Figure 13, we note that the projected chip capacity for 1985 falls short by 6 Mbits. The new projection for 10 Mbits is in the 1990 time frame. We also note that Figure 13a illustrates that the price per bit has not decreased as expected. For 1985, the price is approximately $.03 per bit, which is $.02 more than projected; however, the trend is for lower costs.

As the technology progresses, the cost decreases, the access time (currently, 96 Mbits/sec) reduces, and capacity increases, MBM can play a vital role as a supplement to other technologies. Since numerous Japanese companies have taken up where US companies dropped off, the future remains optimistic for this technology.

I. SUMMARY

Bubble memory technology, although it would not be the panacea that many have thought, is suited for certain tasks. Its portability and reliability make it an ideal candidate for those tasks where the tremendous speed is not required, but rather the durable service over a long period of time is required. Such uses include in control machinery, in recorded messages, at remote sites, at

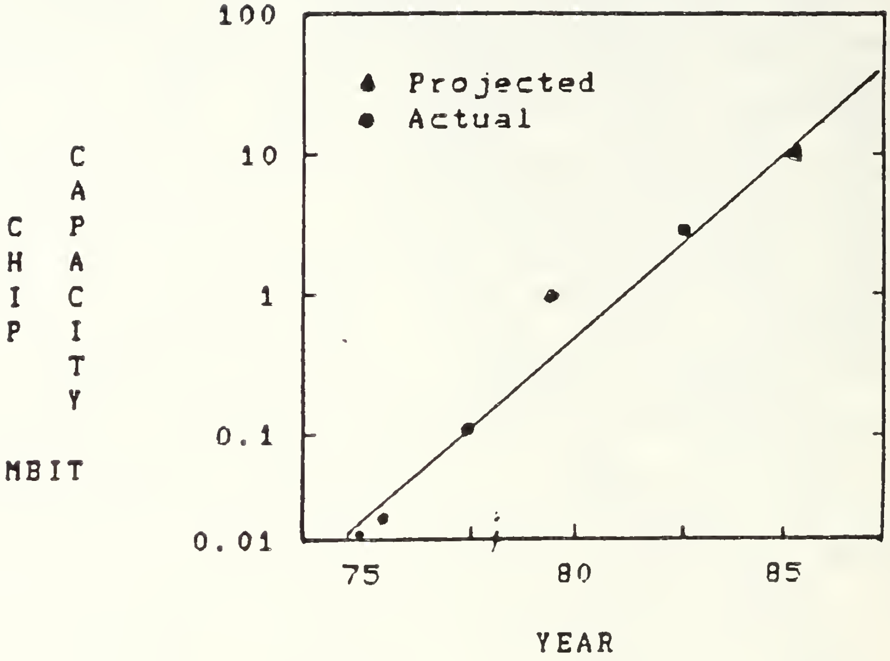


Figure 13. Bubble Memory - Chip Capacity vs. Year.

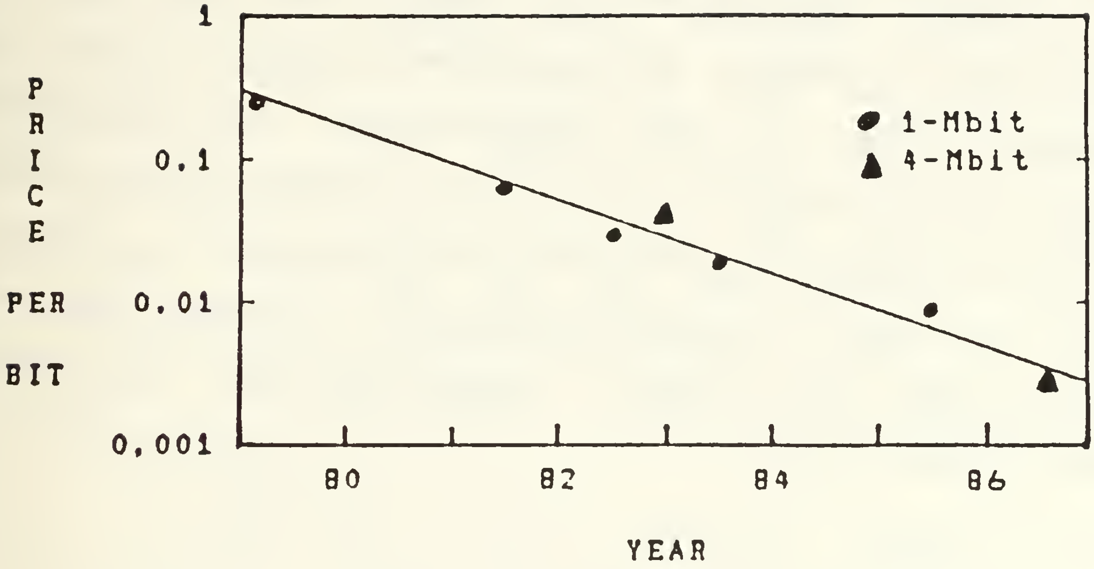


Figure 13a. Bubble Memory - Price/Bit vs. Year.

places where minimum maintenance costs are desired, in instances where vital information cannot be risked, and in memory cassettes or devices which must be transferred over distances.

Military uses of MBM do exist. Intel is devoting considerable research and development effort in MBM, for military usage. It has experimented with enhancing temperature variations from 20 to 85 degrees in centigrade for operational uses. Moreover, when the cost is reduced and the access time is improved, many industrial uses may result. Thus, MBM is still a viable supplement to other storage technologies.

IV. THE VERTICAL RECORDING

The development of the magnetic recording has been a history of pursuing higher recording density. The high density recording is precisely the goal of vertical recording. Several research efforts toward the vertical recording took place during the late 1950's. Since the desired performance had not been achieved, (i.e., the performance of the vertical recording could not match or exceed the conventional recording), the vertical research was abandoned.

In the early 1970's Professor Iwasaki and his coworkers at Tohoku University discovered that the high density recording is inhibited by the well-known effects of the recording demagnetization. This lead to the renewal of research of a practical method of vertical recording. Systematic research on the vertical recording, however, did not start until 1975, and by 1983, approximately 140 reports on vertical recording have been presented in the related field. This eight-year period have seen a slow, but steady, elevation of this subject to the rank of major research on magnetic recording. This trend is expected to intensify in the future [Ref. 12].

A. CONCEPTS, OPERATIONS AND CHARACTERISTICS

The concepts and operations of the vertical recording are similar to the conventional recording. The only difference is that, the key to this new method lies in magnetizing the disk surface material at right angles, i.e., at angles vertical to the surface. In contrast, the conventional recording creates magnetized zones along the surface. With the vertical recording, higher recording densities now span the depth rather than the length of these magnetized regions. Consequently, the raising of the recording density no longer worsens the demagnetizing effect. In fact, the opposite is true. This effect is explained in the following sections. Because the recorded magnetic zones are vertical to the disk surface, higher densities now squeeze their waistline dimensions, rather than their length (see Figure 14).

B. THE ARCHITECTURE

Figure 15 depicts the vertical head being utilized today in the vertical recording. It consists of a main pole made of a thin magnetic film, which is less than 1 um thick, placed vertical to the disk surface, and an auxiliary pole made of a thick ferrite film and located on the other side of the recording medium. On the tip of the auxiliary pole is a coil, which is used for reading and writing. The gap between the these two magnetic poles is less than 100 um.

Reading is performed as the current in the coil induces a concentrated magnetic flux on the main pole. This flux is shown by dashed lines in Figure 15. In the writing process, meanwhile, the magnetic field of the medium magnetizes the main pole and induces a voltage in the coil.

This head is characterized by a strong interaction between the main pole and the magnetic layer of the medium. This operation is carried out by the concentration of the magnetic flux from the main pole into the magnetic layer of the medium. Consequently, only the vertical magnetic field on the tip of the main pole becomes significantly strong. In addition, as the width of the vertical field is governed only by the thickness of the main pole, a purely vertical magnetic field is always applied to the medium regardless of the recording level.

In conjunction with the above vertical head a double layer medium is used. This is done to enhance the reading and writing process tenfold. The magnetic interaction between the main pole and the magnetic layer of the medium is therefore much enhanced.

C. THE VERTICAL MEDIUM

Although the same medium materials as in the conventional recording can be utilized, cobalt chrome (Co-Cr) film is best suited for this type of recording. Co-

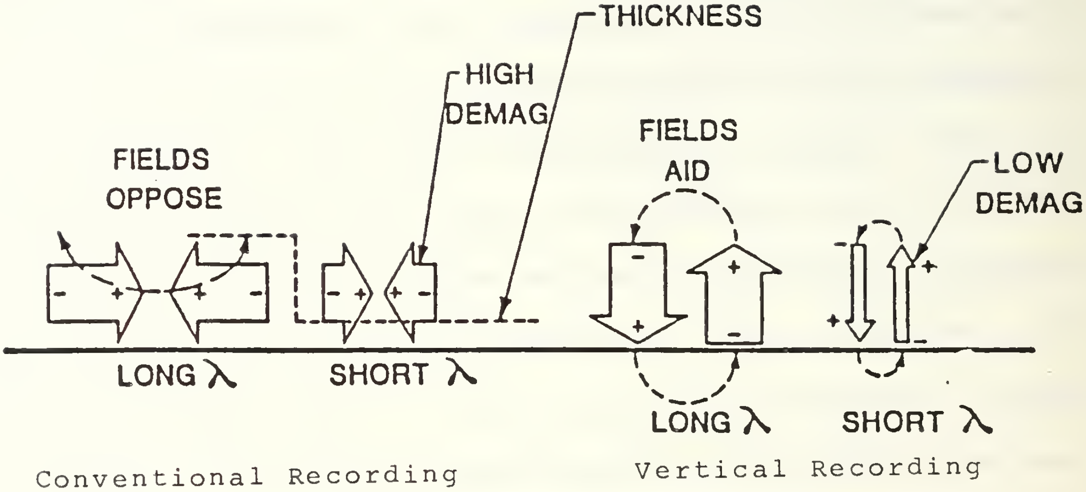

THICKNESS
HIGH DEMAG
FIELDS OPPOSE
FIELDS AID
LOW DEMAG
LONG λ
SHORT λ
LONG λ
SHORT λ
Conventional Recording
Vertical Recording

Figure 14. Concepts of the Vertical Recording.

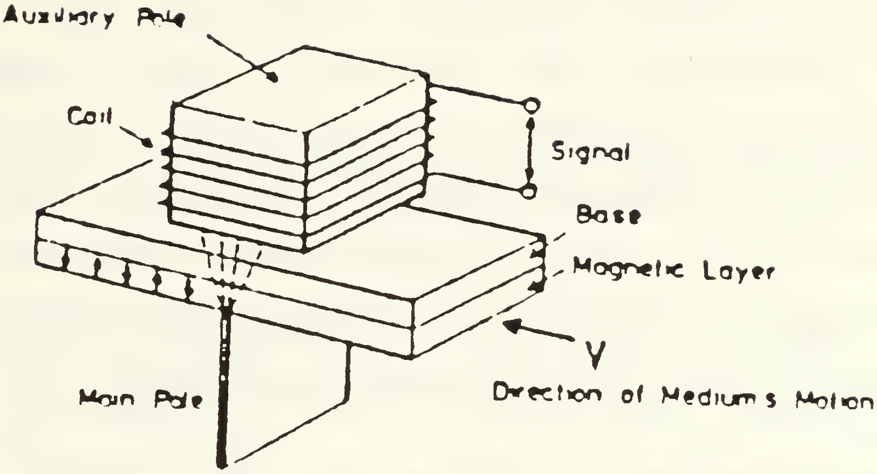

Auxiliary Pole
Coil
Signal
Base
Magnetic Layer
V
Main Pole
Direction of Medium's Motion

Figure 15. Construction of Vertical Head.

Cr film has a wide range of variations, which are not found in the other materials. First, Co-Cr film has the largest vertical anisotropy. Secondly, both Co and Cr are soluble in a composition where the ferromagnetism may appear, making them more controllable with a cohesiveness from 100-2200 Oe. Finally, the Co-Cr film has the distinctive feature that it is composed of closely packed columnar particles. These particles are physically small enough and sufficiently independent of one another magnetically to permit the ultra-high density recording. This columnar structure is not found in other medium materials. Thus, Co-Cr double layer film is currently the leading candidate for the vertical recording medium [Ref. 12].

D. PROPERTIES OF THE VERTICAL RECORDING

In the vertical recording, the adjacent magnetized regions are in anti-parallel states; thus, an attractive force exists between each pair of residual magnetization regions, making them stable. Therefore, a sharp magnetization transition (that region that is subject to demagnetization) can be obtained even in the high-density recording without being affected by the demagnetization. There is no limitation due to the demagnetization imposed on the recording density for the vertical recording [Ref.

13]. This high-density recording can be achieved simply by using a thinner main pole.

E. ADVANTAGES OF THE VERTICAL RECORDING

With the prospects of the vertical recording becoming a reality, there is a great deal of discussions on the future of the conventional recording. The vertical recording offers the following advantages as compared to the conventional recording:

(1) greater linear density (the vertical recording has 100,000 flux reversals per inch, as compared to 15,000 flux reversals per inch, for the conventional recording),

(2) greater areal density (the vertical recording has 10 to the 10th flux reversals per square inch, as compared to 165 times 10 to the 6th flux reversals per square inch for the conventional recording),

(3) thicker medium (for vertical recording, the medium may be thicker than the ones for the conventional recording, since bits are recorded vertically to the medium),

(4) reduced demagnetization (as the lambda gets shorter for the vertical recording as depicted in Figure 14, the adjacent regions are in close, opposed fields, making demagnetization difficult, whereas in the conventional recording, the adjacent regions are still far apart in opposed fields, making demagnetization easy), and

(5) small transition length (it is so small that it is close to zero, for vertical recording).

F. DISADVANTAGES OF THE VERTICAL RECORDING

The future development of the vertical recording will require extensive investigations on new heads and media. Only by developing new heads and decreasing the cost to manufacture media for the vertical recording, can we fully exploit the successful application of this technology.

G. FUTURE TRENDS

Vertical recording is being developed mainly by an alliance of Japanese industry and universities. In this country, the Magnetics Research Laboratory at the University of Minnesota is seriously pursuing the potentials of this new technology. The Vertimag Systems Corporation is the only company in the United States reportedly involved in the vertical recording. The Japanese, on the other hand, have a massive effort going on in the vertical recording. In 1982, the first International Symposium on the Vertical Recording was sponsored in Japan. Of the 23 papers presented on this topic, only three were by U.S.A. authors and all three were from Vertimag. The other 20 were by Japanese authors.

Virtually, every well-known Japanese electronics company is working on this technology. These companies include: Hitachi, Toshiba, Fujitsu, Nippon Electric Company, Sony, Matsushita, and a number of smaller organizations. The announcement of a 3 1/2-inch, vertically-oriented prototype floppy disk in 1983, represents the level of Japanese achievement and dominance in this field. They anticipate production now.

Once a medium is available at a mass-production price and the technology is well understood, there will be a rapid movement into this field by companies in the U.S.A.. The rate of development and market penetration is likely to be constrained for the near future, because of the slow and expensive process to fabricate the media, the large capital investment for the sputter system, and requirements for a new type head. It is expected that the vertical recording may complement the conventional recording for at least the next ten years. The vertical recording, as the Japanese have already realized, represents the next level of magnetic recording technology for the not-too-distant future.

V. THE OPTICAL RECORDING

A. AN INTRODUCTION

In today's society, the expansion of our knowledge has generated data in ever-increasing volumes and given rise to the need for their efficient long-term storage. Storing (writing) these data require economical, compact, and high-speed mass memory systems. Retrieval (reading) of these data require the random-access capability to the selected data.

Over the years, the manufacturers of conventional storage devices have been able to increase storage capacities to keep pace with the growth in data storage and retrieval requirements. However, even more dramatic advances in storage capacity are needed to satisfy these newly emerging requirements. Although, the conventional recording has much room for future growth, i.e., doubling its capacity every 30 months, it is an evolutionary developments, rather than dramatic leaps in increasing storage capacity.

An attractive new technology to satisfy this high-capacity data storage needs may be the optical recording, which makes use of a highly focused laser beam. Research and development of this high-density optical data storage actually began over 20 years ago, with the invention of the laser. The term laser is an acronym for light amplification by stimulated emission of radiation, or a light amplifier. The process of stimulated emission can be described as follows: When atoms, ions, or molecules absorb energy, they can emit light spontaneously, as in the case of an incandescent lamp. A light wave may be used to stimulate the emission. Thus, the stimulated emission is the opposite of stimulated absorption, where unexcited matter is stimulated into an excited state by the light wave. If a collection of atoms is prepared so that more are initially excited than unexcited, then an incident light beam stimulates more emission than absorption, and there is the net amplification of the incident light beam. This is the way that the laser amplifies [Ref. 14].

Like the conventional recording, the optical recording encompasses a family of configurations that address the many requirements of data storage users. In the optical storage technology three configurations exist: read only, write once, and erasable recording. Principles of operations, the architecture, applications, technological implications, media types, the capacity, the cost, future trends and problems of the optical recording will be discussed in the following sections.

B. PRINCIPLES OF OPERATIONS

Figure 16 depicts the write and read operation for optical recording. First, the process for the write operation is reviewed. In the write operation, the drive focuses a high-power laser beam on the underside of the disk and into the preformed track (pre-embossed data pits). The beam passes through the disk substrate (i.e., polymethylmethacrylate, PMMA, which is an absorbing layer), and strikes the thin metal coating (i.e., aluminum reflective layer), and heats the coating. The coating consequently becomes soft. The heat energy is then transferred to the PMMA substrate which generates gases when the substrate has been heated. These gases push up on the metal layer to create bubbles, which are approximately .6 um. Thus, data is recorded.

The process for reading data is more straightforward and simple. In the read operation, a low-power laser beam detects the presence of bubbles by measuring the changed intensity of the reflected light from the disk surface. Thus, data can be read.

C. THE ARCHITECTURE

Figure 17 illustrates a simple optical disk memory architecture. It employs the laser light to write data by burning holes in the medium on a spinning disk. The laser is used both for reading and writing. Only the intensity of the laser beam is different.

The optical disk architecture works as follows. First, the laser emits a beam of coherent light that is broken by a diffraction grating into essentially three parallel beams. Two of the three beams are later used for detecting tracking errors. The third beam, which is the strongest of the three, is the main reading beam. These three beams, moving alongside each other, are then focused by a collimating lens. The beams then pass through a special wollaston prism or polarizing beam splitter (PBS), which allows the vertically polarized projection beams to pass directly through, but, separates the reflected light. The projected beam continues through a quarter wavelength retardation plate, which brings the light back into focus. This changes the polarization characteristics of the beam which is then directed by a tracking mirror and finally focused onto the disk by the objective lens, thereby allowing writing or reading to occur.

If the process is to read, on the return trip, the reflected light retraces the path to the retardation plate. This modifies the polarization, allowing the prism to bend it at right angles to the projected beam and prevent any type of feedback into the laser. Then, the cylindrical lens focuses this separate reflected beam, which falls on a photo receptor array, which in turn is composed of photo

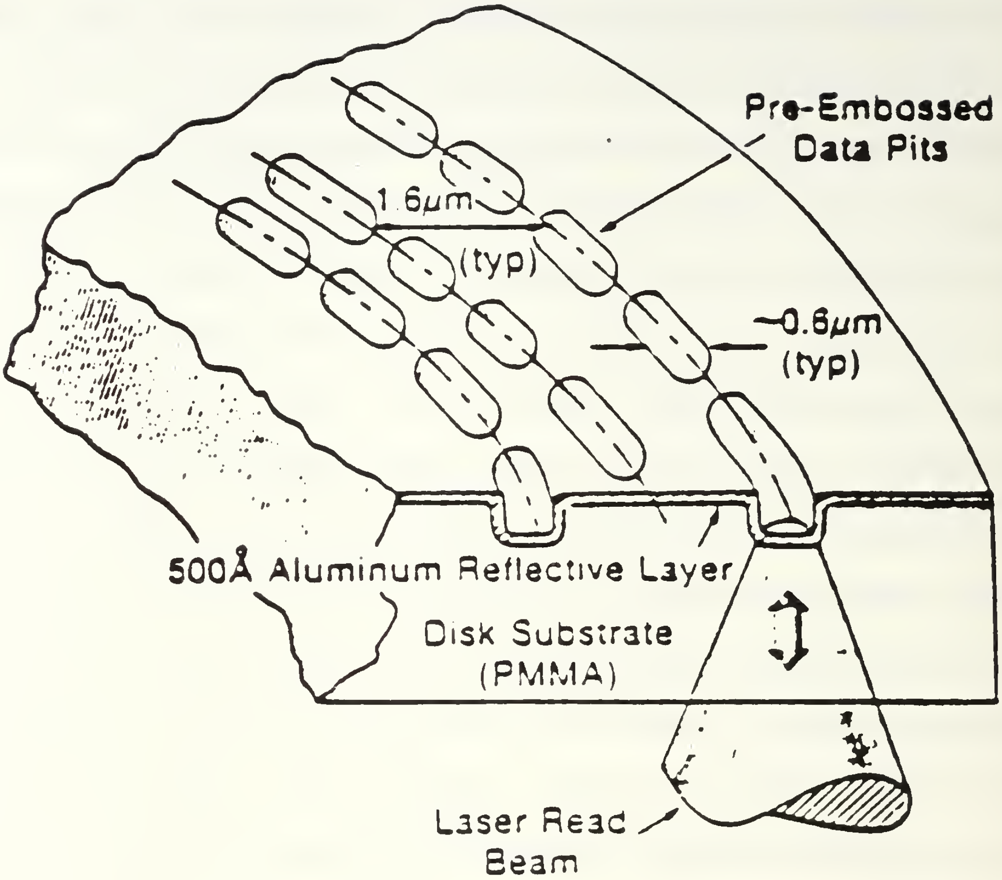

Pre-Embossed
Data Pits
1.6μm
(typ)
0.8μm
(typ)
500Å Aluminum Reflective Layer
Disk Substrate
(PMMA)
Laser Read
Beam

Figure 16. The Read/Write Operation of the Optical Disk.

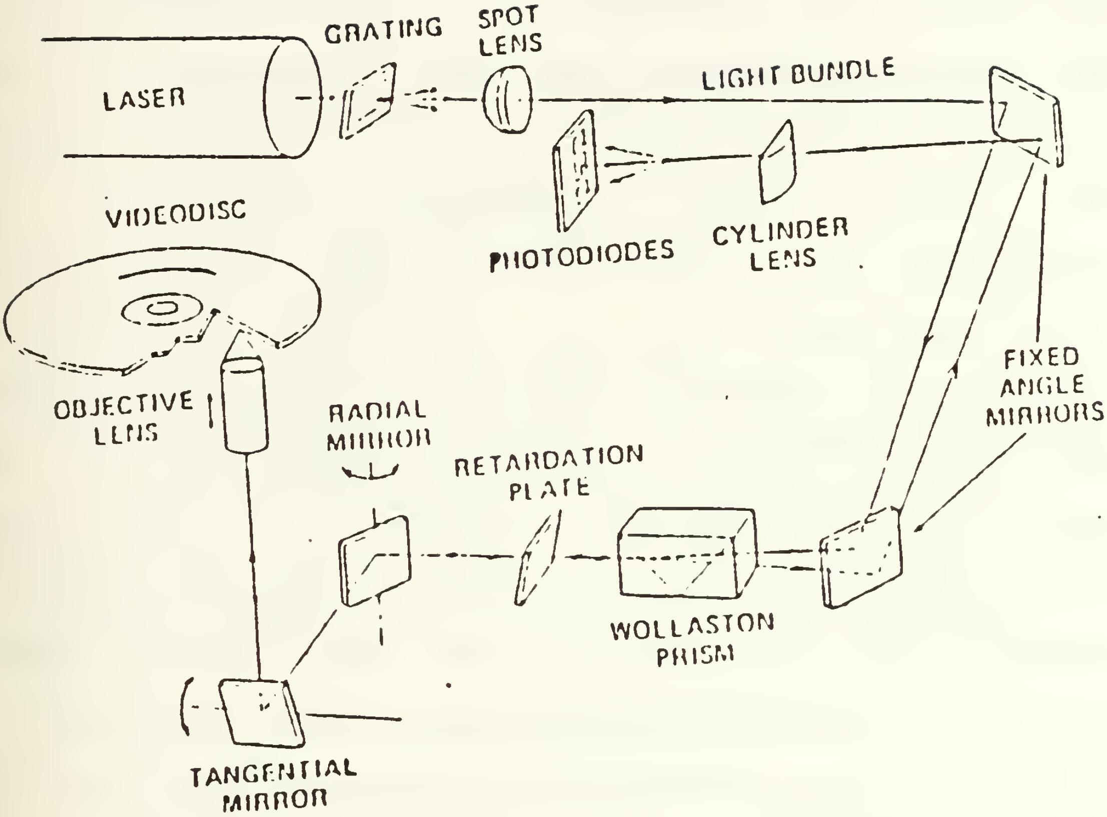

LASER
GRATING
SPOT LENS
LIGHT BUNDLE
PHOTODIODES
CYLINDER LENS
VIDEODISC
OBJECTIVE LENS
RADIAL MIRROR
RETARDATION PLATE
FIXED ANGLE MIRRORS
WOLLASTON PRISM
TANGENTIAL MIRROR

Figure 17. The Architecture of an Optical Disk.

diodes. The function of the photo receptors, which control tracking and focus, is to read directly the variation in beam intensity, which encodes the digital data on the disk.

The two weaker tracking beams and the primary laser are focused by the objective lens on three different spots on the disk (see Figure 17a). The intensity of the two reflected tracking beams is compared by separate areas of the receptor array. Differences between them are interpreted as tracking errors, which are corrected by the tracking mirror. On the other hand, the focus is controlled by detecting changes in the shape of the primary beam. When the disk is in focus, the cylindrical lens will project the reflected beam as a circle on the array of four photo diodes (see Figure 17b). When the disk moves closer or further from the objective lens, the projection becomes elliptical, with more light falling on one diagonal pair of receptors. This difference is detected as focus error and a servo mechanism adjusts the objective lens [Ref. 15].

D. APPLICATIONS OF OPTICAL RECORDING

Applications for optical recording are similar to those of the conventional recording. The difference is that the optical recording has greater capacity and lower cost. However, the optical recording has a longer access time. Also, there are other advantages and disadvantages, which are discussed later. The following are some applications of the optical recording:

(1) archive applications,

(2) reference-file applications,

(3) backup for conventional disk files,

(4) collections of large sets of raw operational data,

(5) large relatively stable conventional files previously saved on conventional disks,

(6) file versions or snapshots of files,

(7) very high-density storage,

(8) removable media,

(9) large capacity per media unit,

(10) permanent, nonerasable, nonmodifiable storage,

(11) fast sequential data recording capability,

(12) fast sequential data retrieval capability,

(13) moderately fast direct-access data retrieval capability,

(14) high level of data integrity, and

(15) low cost of on-line storage.

Figure 18 illustrates five specific applications of the optical recording [Ref. 16].

E. CURRENT OPTICAL-RECORDING STATUS

Figure 19 depicts the three categories of optical recording as well as their capacity, applications, and media and drive costs [Ref. 17]. Figure 19a illustrates the

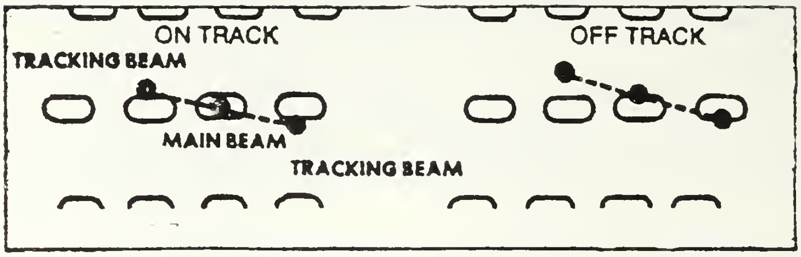

ON TRACK
OFF TRACK
TRACKING BEAM
MAIN BEAM
TRACKING BEAM

Figure 17a. Tracking Error Detection.

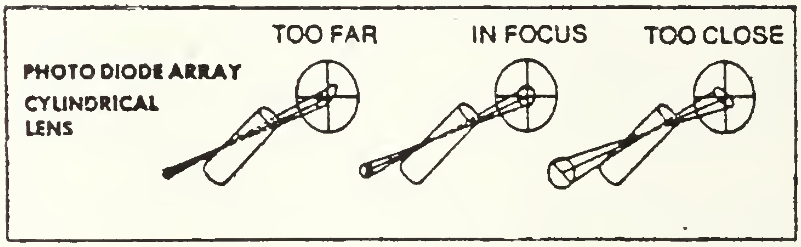

TOO FAR
IN FOCUS
TOO CLOSE
PHOTO DIODE ARRAY
CYLINDRICAL LENS

Figure 17b. Focus Error Detection.

two types of read-only optical data storage disks currently available, along with their characteristics.

The CD-ROM (compact-disk, read-only-memory) disk can now be manufactured in quantity, and thus, can become a medium for the use of large textual databases. Their use can obsolete such references as telephone directories, law libraries, medical histories, book references, and library catalogs. Archival database services can sell their complete historical data on a few CD-ROM disks.

Unfortunately, applying compact disk technology to computers is not as simple as one may think. One of the greatest hurdles is standardization. Although the data on CD-ROMs is organized in a standard way, a standard hardware interface between the players and personal computers has yet to emerge. A hardware interface standard is essential, because audio CD players are designed to transfer data serially, while most personal computers use a parallel scheme for communication with disk drives. Settling on a standard hardware interface will also allow the creation of the operating system for CD-ROM.

The Small Computer Systems Interface (SCSI) is one, though not the only, proposal for standardization. It is based on the Shuggart Associates System Interface (SASI), which is already used for hard disks in personal computers. Other proposals include the IEEE-488 bus and high speed RS-232 serial transfer.

The following applications are more suited to optical storage than to magnetic because:

Magnetic Disk	- $/megabyte too high - Volume of data too large - Not portable
Magnetic Tape	-Physical storage space too large -Capacity per reel too low -No direct access -Media life too short
Optical Storage	-Very high capacity -Low $/megabyte -Direct access -Portable -Long media life

APPLICATION 1

Application:	Extremely large quantities of digital data
Industry:	Energy Exploration - Seismic data - Well information - Satellite data
Benefits:	- Protect data; value increases through time - Increase productivity of technical staffs - Efficient decision making; increase profit

APPLICATION 2

Application:	Storing and retrieving images produced by nuclear and diagnostic medical equipment
Industry:	Medical - Patient information - Diagnostic procedures
Benefits:	- Protect data; X-rays and tests - New diagnostic methodologies - Accurate decisions; life saving

APPLICATION 3

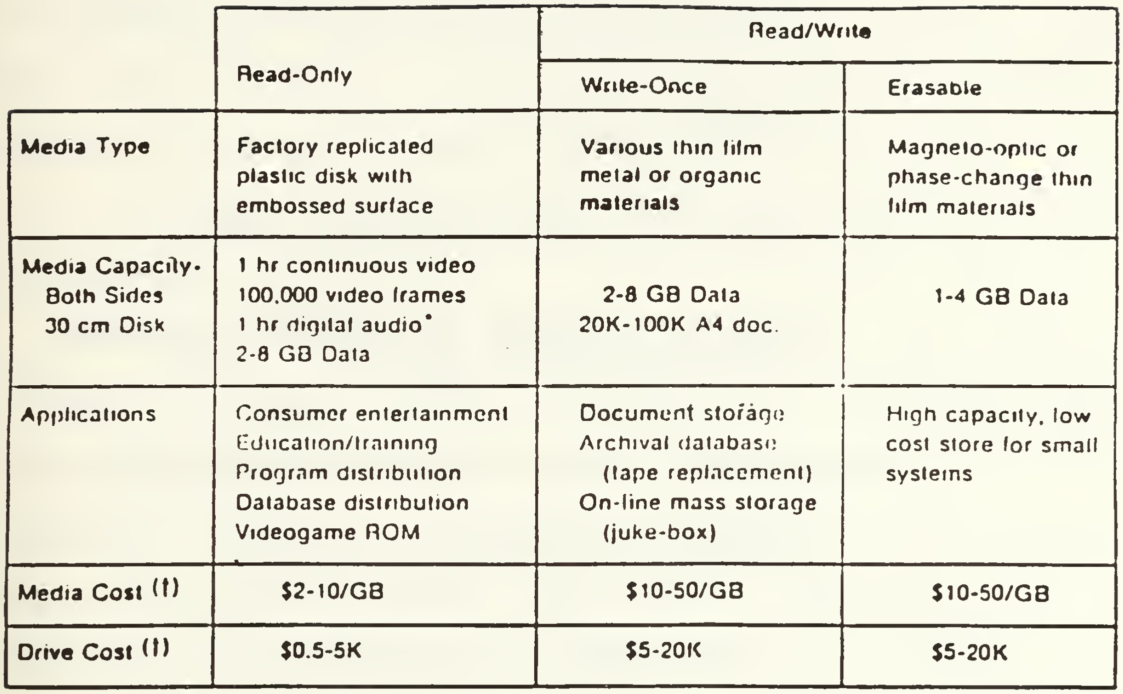
Application:	Storing and distributing large reference files (Books, Periodicals, Catalogs, Abstracts)
Industry:	Libraries - University - Law - Retail catalogs
Benefits:	- Protect data; case histories, abstracts - Distribution; mail platters - Increase efficiency

Application:	Storing, retrieving and distributing images (Maps and Engineering drawings)
Industry:	Manufacturing/Government Topographic maps Drawings - Road maps - Weather maps
Benefits:	- Protect data; track change through time - Distribution - Cost reduction

APPLICATION 5

Application:	Office automation - Document storage
Industry	All - Single electronic copy - Electronic file cabinet - Electronic mail
Benefits:	- Reduce cost, replace paper - Increase productivity - Efficient decision making

Figure 18. Five Applications of the Optical Storage.



	Read-Only	Read/Write	
		Write-Once	Erasable
Media Type	Factory replicated plastic disk with embossed surface	Various thin film metal or organic materials	Magneto-optic or phase-change thin film materials
Media Capacity-Both Sides 30 cm Disk	1 hr continuous video 100,000 video frames 1 hr digital audio* 2-8 GB Data	2-8 GB Data 20K-100K A4 doc.	1-4 GB Data
Applications	Consumer entertainment Education/training Program distribution Database distribution Videogame ROM	Document storage Archival database (tape replacement) On-line mass storage (juke-box)	High capacity, low cost store for small systems
Media Cost (†)	$2-10/GB	$10-50/GB	$10-50/GB
Drive Cost (†)	$0.5-5K	$5-20K	$5-20K

*12 cm disk, 1 side

Figure 19. Classification of Optical Data Storage

1. Digital-Audio Disk (DAD. 1983)

   ... 120 mm Diameter Disk - 1 Hr. Play

   ... Sony / Philips Format Standardization

   ... First High Volume Product for Optical Technology

   ... BER < 10

2. Digital-Data Disk (CD-ROM. 1984)

   ... Based on Digital Audio Disk Technology 550 MB Disk Capacity

   ... Sony / Philips Format Standardization

   ... Playback Unit Price : $1500.00

   ... Access Time < 3s. BER 10

   ... Extendable to Low-End Read/Write Systems

Figure 19a. Two Types of Read-Only Optical Data Storage.

Figure 19b illustrates the current write-once optical data storage products, along with their characteristics and the companies who manufacture them. In January of 1986, Optotech, Inc., of Colorado, will introduce WORM (write once, read mostly memory), for personal computers. The ability to write data on the disk once more within the computer is the difference between WORM and CD-ROM. Thus, once data has been written, the device becomes a read-only device. These devices are currently under development, and are to be introduced in the near future into the marketplace.

WORM will be used for internal databases such as end-of-year financial data, inventories, customer lists, parts lists and other large collections of data developed within a personal computer. The Optotech 5984 is the WORM drive designed to interface to the personal computers. Its double-sided 400-megabyte disk offers 800 megabytes of on-line storage. The cost of WORM when volume production begins is about the same as a 10-megabyte Winchester drive, representing a five-fold decrease in the cost per stored bit of data (approximately, $.10 per megabyte) [Ref. 19].

This write-once data storage disk is approximately one and one-half years ahead of the multiple-write optical data storage disk. Recently, Verbatim Inc. of Sunnyvale, California, announced the successful completion of the first of three development phases of a multiple-write optical data storage disk. The development of acceptable media could lead to relatively rapid introduction of this type disk. The anticipated introduction date is early 1987. The major candidate is magneto-optic recording, which is discussed in the next chapter.

F. MATERIAL REQUIREMENTS

The design and fabrication of optical data storage media are seen as the most critical factors in determining the ultimate usefulness of high-density optical data storage. For data-processing applications, this fact reflects the current status of all three media classification, although to different degrees.

This optical media must deal not only with the generic issues of high-density media characteristics such as media resolution, noise, microdefect and integrity, but must also meet some basic requirements of material properties that are unique to optical data storage, such as good reading and writing capabilities, and an acceptable data rate. Finally, of major importance, there are media lifetime and fabrication cost.

Glass was the first substrate that could be prepared with a quality of good surface, high stability and low BER (bit-error-rate). However, this quality was overshadowed by

COMPANY	DISK CAP.*	DISK PRICE	DATA RATE	ACCESS TIME	SYSTEM PRICE
STC	4 GB	$140	3MB/s	85 ms	$130K
HITACHI	1.3 GB	$300	0.44 MB/s	250 ms	$1000/mo.
NEC/3M	1.3 GB	$250	0.8 MB/s	450 ms	$13,500
OPTIMEM	1 GB	$100*	0.63 MB/s	150 ms	$6000**

* Per Side

** OEM Quantity

Figure 19b. Current Write-Once Optical Data Storage Products.

its cost, bulk, and fragility. Glass is an excellent candidate for archivability.

Polymers, on the other hand, low surface quality due to the molded surface's design. Therefore, a careful design and process control are needed to achieve good surface quality. The ability to directly pattern the surface with positional reference data as wellto provide low cost and high stability, present significant motivation to develop the required fabrication technique and control. The mass production is already underway.

The aluminum disk substrate used in wincnester drives represents an intermediate cost alternative to glass and polymeric substrates, while offering excellent dimensional and chemical stability. However, a spin coated surface layer must be used to achieve good surface quality. Also, the format and positional reference data must be "burned in" after the disk fabrication, which is a time consuming and potentially costly procedure. Aluminum alloys are the best candidates for high performance [Ref. 20].

Currently, the tellurium-based alloy appears to offer the best combination of aforementioned properties, while including sensitivity adequate to meet the necessary requirements. Chen et al. [Ref. 21] have reported a 25% improvement in writing sensitivity for tellurium-based (Te) materials relative to polymers. Te-based alloys are low melting materials, that are easily sublimable, decomposable or vaporized by the laser heat. Also, the Te alloys has adequate archivability (over 10 years). However, due to the high cost of fabricated Te-alloy film (sputtering is employed), and the nature of its properties, (i.e., it must be handed very carefully, since it is poisonous), polymers are going to be the leading candidates for optical recording media.

G. FEATURES AND BENEFITS OF OPTICAL RECORDING

Figure 20 illustrates features and benefits for optical data storage recording [Ref. 5]. A very important feature not depicted is that the media in optical recording is encapsulated; that is to say, it is protected from contamination. The main function of encapsulation is to keep particulate matter away from the plane of focus at the information storage layer in order to minimize its effect on reading quality, particularly BER. A second function is to shield the storage layer from potentially corrosive materials, such as water vapor in the surrounding of the disk. A third function is to protect it from user abuse, whether the abuse is intentional or not.

H. LIMITATIONS

The total data storage density (the areal density) is the product of the lineal data density along the recorded track and the track density in the radial direction.

Optical recording has the following inherent limitations: lineal density - 300K bits per inch (BPI), track density - 24K tracks per inch (TPI), and hence areal density of 8 * 10 to the 8th track revolutions per square inch (TRPI).

the lineal density is limited by the readout or playback step of the optical recording, since the finite resolution of the read beam results in rapidly diminishing playback signal amplitude. In order to accomplish the above lineal density, the demands on disk flatness and focus servo performance can indeed be very challenging. The track density is limited by the finite diameter of the read beam, which results in an increased crosstalk signal (external noises) from the adjacent tracks as the track separation is reduced [Ref. 22].

I. FUTURE TRENDS

The trends in the optical recording are three. The first trend is to improve the areal density. This can be achieved by the development of an enhanced servo control system. This is essential, since a high degree of accuracy is required between the position of the head and the location of the data on the media surface.

The second trend is to improve the data transfer rate. This can be accomplished by the use of integrated arrays of lasers. The multiple, independently modulated output beams of the array are focused within the field of view of a

Features	Benefits
- Increased on-line capacity for optical designed applications	- Lower cost per on-line MB - Justifies keeping must larger amounts of data on-line - Have files on-line for applications which have a high payback but which are not now computerized
- Removability	- Lower cost per off-line MB of media storage - Store data that is presently not machine readable - No staging - Portability of large amount of data from on-line to off-line and vice versa - Remote distribution of large files
- Long Media Life	- Reduced handling and exercising costs - Fewer media errors in stored data
- Random Access	- Because of density and removability, now able to access randomly files that were cost prohibitive on magnetic disk - Improved information retrieval
- Non-Alterable	- Protection of permanent data; no chance of accidental erasure or change
- Volumetric Efficiency (MB per physical cu.in.)	- Reduced physical storage costs - 93%
- Media	- Reduced Number of mounts at least 40:1
- Drive	- Reduced floor space per on-line MB for drives
- Hardware	
- 3MB/SEC Data Transfer Rate	- Faster throughput
- Outboard indexing	- Improved channel utilization
- Track buffer	- Improved channel utilization - No RPS miss
- One control module per drive	- No control module sharing - Faster throughput
- RAS	
- Error checking when data written	- Better data integrity
- Error checking when data read	- Better data integrity
- Optical heads positioned a thousand times higher than magnetic heads	- No head crashes - No head wear - No media wear
- Error logging of soft errors	- Know when to copy platters for data integrity

Figure 20. Features and Benefits of Optical Storage.

single objective lens, and the data are recorded and retrieved in parallel from several adjacent tracks at the same time. The use of the laser array should permit an extremely high data rate (> 10 MB/s) to be achieved without placing excessive demands either on disk rotation velocity (and therefore servo performance) or on the output power levels of each individual laser within the array.

The third trend is to increase the signal-to-noise-ratio (SNR). This can be accomplished by increasing the read beam power proportionally to the critical threshold power that may damage the track. This increase in the read power results each time that the data rate is increased. Increasing the data rates requires a corresponding increase in the read beam to maintain SNR. This can be accomplished by utilization of more suitable storage medium [Ref. 22].

Figure 21 depicts the future trends for the optical recording, and includes hard and soft magnetic disk storage technology for comparisons. For each technology, there are two columns of data. The left column of figures represents the maximum, while the right column of figures is the minimum.

J. THE SUMMARY

The prospects of the optical data storage have continued to strengthen and grow during the past few years , as significant developments occurred in almost every facet of this technology. Optical storage has received a tremendous impetus from the introduction of the Sony CD, digital, audio disk. This consumer item not only has created a broad acceptance of optical disk devices, but, being a high volume product, has created mass-production components applicable to digital storage devices.

The optical data storage continues to demonstrate the potential to become an important factor in the field of high-capacity on-line storage for databases. Read-only and write-once technologies, which are largely complementary to the existing conventional storage, are already emerging into the marketplace. The erasable disk technology systems continue to demonstrate progress, and may one day be an alternative to conventional magnetic recording devices.

	Hard Mag Disk		Soft Mag Disk		Optical Disk	
Data Transfer Rate (MBit/Sec)	100	24	100	24	100	24
RPM	19000	4700	9500	2400	9700	2400
TPI	3000	1200	1500	750	38000	38000
Platters	4	4	1	1	1	1
Access Time (ms)	7	15	20	40	15	30
Capacity (GBits)	5	2	1	.5	15	12
Removable	No	No	Yes	Yes	Yes	Yes
Cost	24	4	5	1	30	12

5 1/4 - INCH DISK

Figure 21. Future Trends of Optical Disks vs. Magnetic Disks.

VI. THE MAGNETO-OPTIC RECORDING

A. AN INTRODUCTION

The main developmental thrust in multiple-write data storage (i.e., erasable optical data storage) is the magneto-optic recording (MOR). As its name implies, the magneto-optic recording is a combination of conventional magnetic and optical technologies. This recording has been around for over fifteen years; but due to unsuitable media, it has remained dormant until recently. However, the introduction of newer, rare-earth transition metal (RE-TM) films, which possess vertical anisotropy (and hence, the magnetic domains are normal to the film plane), has placed the magneto-optic recording research and development into the forefront.

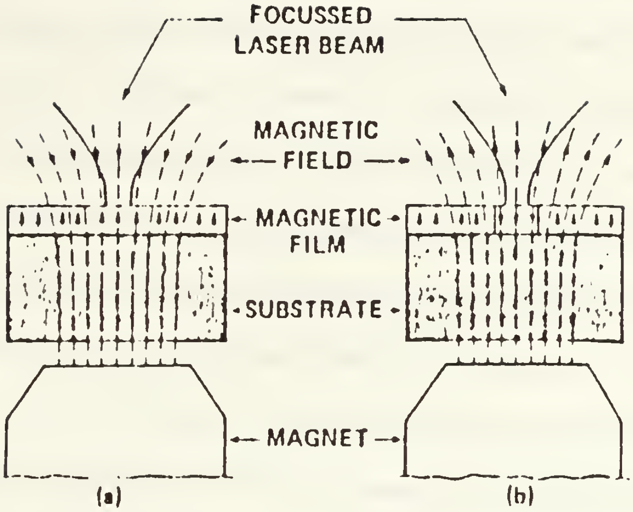
The MOR process is based on two well-known physical phenomena, the Curie effect and the Faraday effect. The Curie effect involves raising the magnetic material to a specific temperature, where the material is most susceptible to magnetic change (i.e., it is demagnetized). The Faraday effect is the change of rotation of polarized light as the light passes through a magnetized medium. The light can rotate left or right, according to the direction of magnetization. In effect, when the light is reflected from a magnetic surface its polarization is changed to reflect the magnetization of the surface.

MOR has advanced to a stage where it has created a surge of enthusiasm. This chapter includes its principles of operation, its architecture, and its technological implications, which include media, features, benefits, limitations, and future expectations.

B. BASIC OPERATIONS

The recording process in magneto-optic films requires the simultaneous application of a bias (externally applied) magnetic field ti be directed oppositely to the initial film magnetization, together with a localized heat pulse to be supplied by the focussed recording laser beam. Figure 22 shows schematically how MOR works [Ref. 23]. A beam of light from a laser is focussed onto the surface of the perpendicularly thin film causing the film to increase in temperature, to the Curie point, in the area of the laser beam. The localized increase in temperature causes a localized decrease in coercivity, thereby allowing the bias field, which is applied antiparallel to the original magnetization direction, to reverse the direction of the magnetization in the heated region. On cooling, the reverse magnetized domain persists. Thus, the writing occurs.

The same magnetic head is used both for the writing and the reading. The reading is accomplished with a lower power


FOCUSSED
LASER BEAM
MAGNETIC
FIELD
MAGNETIC
FILM
SUBSTRATE
MAGNET
(a)
(b)

(1a) A focussed laser beam raises the local temperature of the medium so that the applied magnetic field is able to write a reversed domain. (1b) The domain is erased by the same process, now aided by an oppositely-directed magnetic field.

Figure 22. The Magneto-Optical Recording Basic Operations.

laser beam utilizing the Kerr effect. The Kerr effect results in a small rotation and some ellipticity being introduced into the reflected component of the read beam. Thus, when light is directed to a magnetized surface, the polarity of the light beam changes slightly. Upon being reflected back, the light whose polarity has changed due to the effect of the magnetization of the magnetized area, rotates slightly. This rotation is nearly undetected, from .05 to .3 of a single degree, but it is enough to be read by an optical device.

The erasure process is essentially equivalent to that of the writing process, except that the direction of the externally applied bias field is reversed. Due to speed limitations on switching (i.e., changing direction of the current) the relatively large magnetic field, one revolution is used to erase the sector (i.e., set the magnetization to the zero direction) and a second revolution is used to write the ones on the disk.

C. MATERIAL REQUIREMENTS

Amorphous, rare-earth, transition-metal(RE-TM) thin films are the most widely used media for MOR. MOR storage techniques in these amorphous RE-TM films have the advantages of high-bit density and contactless write, read, and erasure operations. High-bit density is accomplished via the storage of data in a sequence of magnetic domains, while writing and erasure is performed by a local temperature rise in conjunction with a low external magnetic field. Although such materials are erasable and rewritable, they can still achieve recording densities comparable to those of write-once optical disks. These materials are ferrimagnetic in behavior, that is, the magnetization persists even when the applied field is reduced to zero (i.e., it possesses a spontaneous magnetic moment). In gadolinium-cobalt (GdCo), GdFe, and terbium iron (TbFe), for instance, the magnetic moment of the rare-earth atoms (Gd or Tb) aligns antiparallel to the magnetic moment of the transition metal (Co or Fe). Since the temperature dependence of the two rare-earth and transition-metal magnetizations are different, it is possible to produce alloys which exhibit a temperature where the rare-earth and the transition-metal magnetizations are equal and opposite so that the net magnetization goes to zero [Ref. 24].

These RE-TM materials are also quite stable against applied fields and at moderate temperatures. They have high cohesivity, and they can be used for archival storage, although more testing is necessary to ensure greater-than-five-year archival storage.

There are various combinations of RE-TM elements suitable for MOR. Gadolinium-cobalt films were studied early on for this application but Imamura determined that it was easier to stabilize magnetic domains in GdTbFe (Gadolinium Terbium Iron) films [25]. This marked the beginning of the use of ternary alloys in order to optimize the magnetic and magneto-optical properties. GdTbFe films have remained the most popular films for MOR. The advantages of GdTbFe films are that they are amorphous, ferrimagnetic, have good Kerr rotation, and moderate Curie temperature. Thus, RE-TM films have the performance and stability required for MOR.

There are also other classifications of films which are, although suitable, not entirely optimum for MOR. These are the polycrystalline films, of which CoFe is the most common. These films have higher conesivity than the RE-TM films as well as better SNR (35 decibels - a decibel, dB, is a unit for measuring the relative loudness of sounds, from a range of 1 to 130, for CoFe as compared to 45 dB for GdTbFe - The optical media storage guidelines is 45 dB). This results in lower BER. But these films have many problems of which stability is the major. Due to their intrinsic properties, these media are relatively unstable.

D. FEATURES AND BENEFITS OF MOR

The MOR technology is an attempt to abstract from the conventional recording the experience and know-now and from the optical recording the high-density capacity and low cost. The following are some of the major features and benefits of MOR:

(1) as media RE-TM alloy are the leading candidates,

(2) multilayer interference coatings are important to achieving adequate SNR,

(3) the cyclability is good, greater than 10 to the 7th cycle reported,

(4) the data retention is good,

(5) the drive technology is in place today,

(6) an overwrite requires a sequential erasure, followed by a re-write,

(7) the stability of the RE-TM films is adequate,

(8) it has the similar performance of the optical recording and the erasability of the conventional recording,

(9) it has removable/portable capability, and

(10) the nondestructive readout is achieved by the Kerr effect.

Figures 23 illustrates three of the major companies who are developing MOR products, along with its salient characteristics. Although IBM is not included in Figure 23, it has recently joined the quest in the development of magneto-optic disks. The leading contenders are currently Sharp/Verbatim, 3M, and Matsushita [Ref. 20].

E. LIMITATIONS

Experiments have shown that the performance of the magneto-optic recording media is currently limited by system parameters, such as laser wavelength, and the

	SHARP	3M	MATSUSHITA
READ/WRITE WAVELENGTH	780NM	820NM	830NM
MEDIA	GDTBDYFE QUADRILAYER	- QUADRILAYER	GDTBFEGE BILAYER
DISK ROTATION RATE	900 RPM	1860 RPM	1800 RPM
DATA RATE	4 MB/S	16 MB/S	4 MB/S
RECORD POWER	4.8 MW	10 MW	9 MW
READ POWER	1.8 MW	1.5-3 MW	1.8 MW
PLAYBACK CNR (30 KHZ)	40 DB	50 DB	49 DB
NO. OF CYCLES	-	-	$>10^7$

Figure 23. Performance Characteristics of Magneto-Optic Media.

numerical aperature (size of the opening) of the focus lens. Since the size of the pits on the media is determined by the division of the wavelength by the numerical aperature and the multiplication of a constant (.56), the best that can be achieved is a wavelength of 820 nanometers, and a numerical aperature of micrometers to 1 micrometers. Hence, optimum values are not utilized [Ref. 23].

Moreover, since the size of the pits, which is one primary factor that determines the bit density along the track, is limited, the density capacity, which is equivalent to write-once optical density capacity is also limited. The other primary factor that determines the bit density is the number of bits per recorded pit. The total disk capacity of an optical disk depends on the number of bits that can be stored on a single revolution (track) of the disk and the total number of revolutions (tracks). The major factors that determine the total number of tracks are the size of the disk and the track-to-track spacing, which must be sufficient to reduce crosstalk between the tracks to an acceptable level, since SNR does limit large pit length.

F. FUTURE TRENDS AND POTENTIAL PROBLEMS

The future trend of the magneto-optic recording is to increase the areal density, which is similar to write-once and read-only technologies, and enhance SNR. Rothchild, president of the Rothchild Consultants Research Firm, claims that SNR can be easily enhanced by utilizing error detection procedures similar to formatting a hard disk. Hence, high error rates, due to low SNR can be eliminated.

Most of the technical difficulties encountered with the MOR technology reside at the media level. A process used to create low volumes of disks in the laboratory under ideal circumstances is not easily adapted for mass production of thousands of disks per hour. Probably, the two major difficulties in mass production is controlling the thickness of the recording layer and guaranteeing the integrity of the written data for a minimum of 10 years. Therefore, a better amorphous substrate material is required. Until the introduction of a better medium, the only way of ensuring data integrity that once data is recorded is to protect the layer from oxidizing. Moreover, a question to ponder is the importance of a greater than 10 year archivability [Ref. 26].

Figure 24 depicts the MOR status. Figure 24a illustrates applications for this technology in comparisons with other technologies.

Magneto-optic drives will be used in a large variety of applications, from memories for small portable computers to mainframes with large on-line databases. Some possible uses include:

(1) an inexpensive replacement for mainframe peripherals (1 to 2 GByte),

(2) an archival back-up for those peripherals,

(3) a replacement for small Winchester drives (.05 to .3 GByte),

(4) an archival back-up for Winchester or for magneto-optic drives, and

(5) an on-line mass storage system that combines removability with random-access and terabyte capacity.

G. SUMMARY

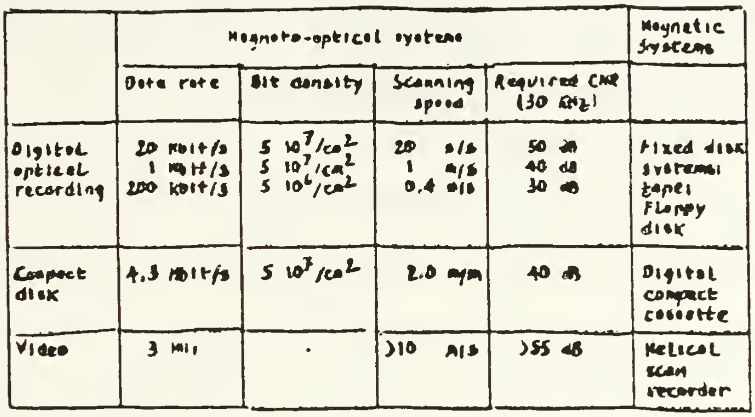
Magnetic-optical memory products are expected to be introduced into the marketplace in 1986. When this does happen, the MOR technology will have an enormous impact on the design of future digital data storage systems, especially personal computers.

Magneto-optic drives will combine large storage density, low cost, random access and erasability with removability. Removability is particularly important because it will enable a drive to be used with any medium, whether it be erasable, non-erasable or read-only. It is very likely that

Company	Media	Density Mbi^2	Data Rate MB/s	CNR dB	R/W Powe mW Inc.
CANON	GdTbFeCo	-	-	45	7/2
DAICEL	GdTbFe	-	-	50	5/1
MEI	GdTbFeGe (P)(PM)	250	0.3	49	9/1-8
NHK	GdTbCo (P)(G)	-	0.75	40-43	10/1
NIKON	TbFe/GdFeCo (G)	-	1.25/0.3	48/55	8.7/3.7
OLYMPUS/KDD	GdTbFe (P)(PM)	250	0.3	48	7/1.1
PHILIPS	GdTbFe	258	0.125	44	8/1
RICOH	TbFeCo	-	-	45	-
SHARP	GdTbDyFe (P)(G)	140	0.50	50	5/1
SONY/KDD	TbFeCo (P)(PM)	320	0.25	52	7/1
3M	RE-TM	300	2.0	>50	12/3
XEROX	RE-TM (A)	-	1.25	56	8/1

P - Pregroove: Pm-PMMA: G-Glass: A-Aluminum Substrates

Figure 24. The Magneto-Optic Status.



	Magneto-optical systems				Magnetic Systems
	Data rate	Bit density	Scanning speed	Required CNR (30 kHz)	
Digital optical recording	20 Mbit/s 1 Mbit/s 200 kbit/s	$5\ 10^7/cm^2$ $5\ 10^7/cm^2$ $5\ 10^6/cm^2$	20 m/s 1 m/s 0.4 m/s	50 dB 40 dB 30 dB	Fixed disk systems, tapes, Floppy disk
Compact disk	4.3 Mbit/s	$5\ 10^7/cm^2$	2.0 m/s	40 dB	Digital compact cassette
Video	3 MHz	.	>10 m/s	>55 dB	Helical scan recorder

Figure 24a. Applications for Magneto-Optical Storage in Comparison with Existing Magnetic Systems.

MOR drives will be multifunctional, thereby allowing software to be sold on read-only disks, back-up to be performed on either erasable or non-erasable disks, and erasable disks to be used on-line for system functions. Inexpensive MOR drives are now practical because of advances in amorphous RE-TM films.

VII. TWO OTHER RECORDING TECHNOLOGIES

Two other recording technologies which are not as prevalent as conventional and optical recording technologies are the RAM (random-access memory) and the Bernoulli cartridge. These technologies, especially RAM, do not have the high-capacity potential and low cost of the conventional and optical recording. However, they do have other superior characteristics. This chapter introduces these two technologies and their operation, characteristics and useds.

A. RAM

In early computer systems, memory technology was very limited in speed and high in cost. Since the 1970's, the advent of high-speed random-access memory (RAM) chips has significantly reduced the cost of computer main memory by more than two orders of magnitude. Chips no larger than 1/4 inch square contain all of the essential electronics to store hundreds of thousands of bits of data or instructions.

Although the RAM acronym indicates the random-access capability, it is actually a misnomer, since almost all semiconductor memories except for a few special types can be randomly accessed. A more appropriate name for this memory would be a read/write RAM to indicate that data can be written into the memory as well as be read out of it randomly.

There are two basic types of RAMs, static and dynamic. The differences are significant. The RAM type refers to the structure of the actual storage circuit used to hold each data bit within the memory chip. A dynamic memory uses a storage cell based on a transistor and capacitor combination, in which the data is represented by a charge stored on each of the capacitors in the memory array. The memory gets the name dynamic from the fact that the capacitors are imperfect and will lose their charge unless the charge is repeatedly replenished (refreshed) on a regular basis (usually every 2 ms). If refreshed, the data will remain until intentionally changed or the power to the memory is shut off. They require supplementary circuits to do the refreshing and to assure that conflicts do not occur between refreshing and normal read/write operations. Although they do have to contend with these extra supplementary circuits, dynamic RAMs still require fewer on-chip components per bit than do static RAMs, which do not require refreshing. Since dynamic RAMs do require fewer components, it is possible for them to achieve higher densities than static RAMs. These higher densities also lead to lower costs per bit. Static RAMs, in contrast, do not use a charge-storage technique; instead, they use either four or six transistors to form a flip-flop for each storage cell. Once the data is loaded into the flip-flop storage elements, it will indefinitely hold this data until it is purposely changed or the power is shut off. Static RAMs are easier to design. They compete well in applications where the memory requirement is not too great, since the cost of the smaller memory is not overwhelming.

There is another trade-off to be made with random-access memories. In addition to the choice of dynamic vs. static types, there is the choice of MOS (metal oxide semiconductor), and bipolar chips. Bipolar devices are faster, and provide better performance, but have not yet achieved the higher densities and hence the lower cost of MOS, as well as its lower power consumption.

In order for the RAM technology to be viable as an alternative to on-line high capacity media storage, this technology must have the capability of high capacity. In terms of capacity, since the early 1970s, when a RAM density of 1K (1024 bits) per chip were introduced, improvements in semiconductor processing and circuits design have made practical an increase in density. This increase went to 4K bits on a chip to 16K bits, and in 1980 to 64K bits. Limited production of dynamic RAMs 256K bits began in 1983.

Samples of dynamic and static RAM devices and prices are as follows [Ref. 27]:

(1) Fairchild has introduced a 45 ns MOS 64K x 1-bit static RAM for $ 90.00;

(2) Integrated Device Technology states that its 64K x 4-bit static RAM will deliver access times on the order of 45 ns and cost less than $ 1000.00;

(3) Hitachi states that its 25 ns MOS 64K x 1-bit static RAM is the fastest available and will cost $ 68.50 (in bulks of 10,000);

(4) Electronic Designs' 16K x 4-bit MOS static RAM has an access time of 55 ns and costs $245.00 (in bulks of 100);

(5) Toshiba intends to market its 45 ns, 64K x 1-bit mos static RAM in Nov 1985 for $ 30.00 each;

(6) Toshiba and Vitelic are both introducing 1 M x 1-bit MOS dynamic RAM at the end of this year. These devices employ geometries of less than 2 microns to attain access times less than 100 ns.

B. RAM CHARACTERISTICS

Some of the characteristics of MOS-based RAM are:

(1) Consumes little power;

(2) Does not usually require back-up;

(3) Very fast, with access speeds below 100 ns;

(4) Ideal for systems that write a lot, but stores little;

(5) Very expensive;

(6) Areal density is 10 to the 4th bits per square inch for 64K bit RAM, and

(7) 256K bit density, with 1M bit density in the wings.

A new type of RAM that is gradually making its mark on the market is the nonvolatile RAM. Nonvolatile memories are a most interesting and active segment of memory technology. These devices retain their contents even when the system loses power. This type of nonvolatile RAM is actually nothing more than the combination of the flexibility of the RAM with the permanence of the ROM (read only memory), when power is removed. The result is that for every stored bit there are two memory cells, one of which is volatile and the other nonvolatile. During normal system operation, the nonvolatile RAM uses the volatile memory array, but when it receives a special store signal, data held in the RAM area is transferred into the nonvolatile section. Thus, the RAM section provides unlimited read and write operations, while the nonvolatile section provides back-up when power is removed. The drawbacks to this almost ideal memory element are twofold. First, it wears out. That is, the electrical process used to store data in the nonvolatile array causes a steady deterioration in the ability of the memory to retain data for a guaranteed period of time. Currently, available capabilities range from about 10,000 to over 1,000,000 write cycles, but many times that number are needed for general purpose use. Second, nonvolatile RAMs have only reached 4K bits, with smaller amounts already finding their way onto single-chip microcomputers [Ref. 29].

C. SUMMARY OF RAM TECHNOLOGY

The possibilities of RAM technology of replacing spinning media is remote, although the microprocessor chip technology continues to improve. Production of 256K chips is revving up earlier this year, and the 512K chip is already a step-child of the much heralded megabit chip, which is being introduced now.

Right now, although large semiconductor electronic memories are available, the cost is prohibitive. Intel's FAST 3825, a 12 MB to 144 MB RAM disk system, which is made up of 64K chips, is priced in the $ 100,000.00 range [Ref. 28]. In the foreseeable future it seems that electronic memories are not close to the cost per megabyte offered by the spinning technologies. For example, utilizing the highest density available today (256K), to put together a 10 MB memory with 256K RAMs. It will eventually get down to about $ 5.00 per chip and a total of 320 chips will be needed at a cost of $ 1600.00 for the chips and another $ 1000.00 to put them all together. A total of $ 2600.00 as compared to $ 350.00-$450.00 for a 10 MB Winchester disk. However, if the density is not important, the cost is of no concern, and the speed is of utmost important, volatile RAM is the best alternative.

D. BERNOULLI CARTRIDGE

In 1981, three IBM employees left Big Blue to start their own company in Utah, the Iomega Corporation, to manufacture what they believed to be the ideal mass storage system disk, the Bernoulli disk. The Bernoulli disk has the best attributes of floppy disks and hard disks without their shortcomings. Floppy disks, for example, trade low storage density and long access times for portability, ease of backup, and low cost. Hard disks, on the other hand, trade sensitivity to dirt and shock for an increase in storage capability and speed.

The Iomega Corporation's Bernoulli disk, the Alpha 10, is an eight inch, cartridge-loaded floppy disk that holds 10 megabytes. The magnetic medium is only a three mil thick, mylar floppy disk. Unlike a normal floppy, the Alpha 10 disk is housed in a magazine-sized plastic cartridge. The cartridge, like a Video cassette, automatically closes up when removed from a drive, which protects the disk from contamination. When the cartridge is inserted into the drive, the disk is exposed to a flat plate over which it will fly (around the) spindle and move close to the read/write head. The drive is given stability and the close head-to-disk clearance crucial to high storage density by taking advantage of the "Bernoulli Principle".

E. BERNOULLI EFFECT

Daniel Bernoulli, a Swiss mathematician, observed more than 200 years ago that the pressure of a moving fluid is indirectly proportional to its speed. If a disk spins close to a stationary surface, a negative pressure is generated between the two and the stabilizing effects cause the disk to fly at a determined distance above the rigid Bernoulli plate. Another application of this principle is in the head design. The read/write head in the Bernoulli disk is stationary and protrudes through a banana-shaped slot in the Bernoulli plate. The head mounting bracket is shaped so it protrudes a few thousandths of an inch above the plate. These "bumps" in the plate cause the secondary area of the Bernoulli effect, drawing the disk even closer to the head, which has a 4 to 7 microinches of clearance.

The advantages of this scheme are obvious. Because the disk flies, rather than the head, disturbance of the device causes the disk to lose lift and fall away from the head, rather than toward it. Hence, the head can not crash with the Bernoulli disk.

Figure 25 illustrates the Bernoulli pumping effect [Ref. 30]. The Bernoulli technique takes advantage of the rapidly flowing air (i.e., the air next to the surface of a rotating disk), high disk rotation speeds, and megabyte data storage of a floppy disk. The rapid rotation of the disk generates an airflow that pulls the disk surface toward the drive read/write head. The shape of the drive head, however, is engineered to prevent the disk surface from actually touching the head. As the disk surface approaches the head, an air bearing of less than ten microinches forms, holding the disk away from the head.

Figure 25a depicts the three types of products along with their characteristics. A single, 10 Mbyte drive costs $ 2,695.00. A dual drive, 20 Mbytes costs $ 3,695.00. When two Alpha 10 drives are installed with a power supply in a box, the result is the Bernoulli Box. Figure 25b illustrates the features and benefits of the Bernoulli disk drive.

F. SUMMARY

For small database systems, the Bernoulli disk drive offers a very viable alternative. The Iomega innovations have created a system with a 24,000 bits-per-inch density, a 300 track-per-inch track density, a data transfer rate of 1.13 megabytes per second, a system latency of 20 milliseconds, and an average access time of 50 milliseconds. The storage of the Bernoulli disk is as/p reliable, quiet, and quick as any winchester disk currently available, and much cheaper. Better yet, the Bernoulli disk provides a credible backup facility for a winchester disk, one that doesn't require the incessant changing of

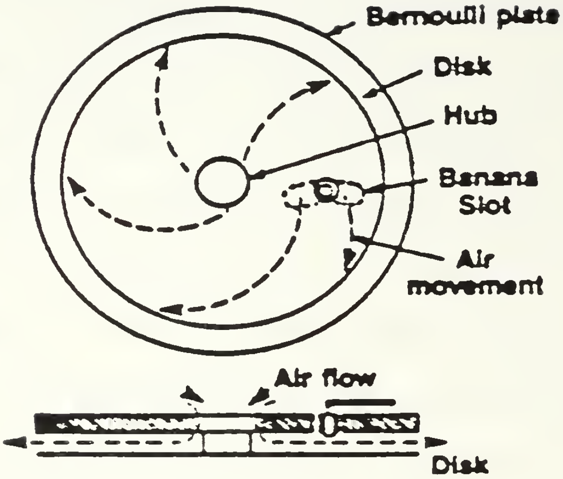

Bernoulli plate
Disk
Hub
Banana Slot
Air movement
Air flow
Disk

Figure 25. The Bernoulli Pumping Effect.

	BETA-5	ALPHA-10	ALPHA-10.5
Drives	5 1/4 in	8 in	8 in
Access Time	65 ms	55 ms	50 ms
Data Transfer Rate	5 MBit/sec	1.13 MBytes/sec	1.13 MBytes/sec
Capacity	5MBytes	10 MBytes	10.5 MBytes

Figure 25a. Bernoulli Box Cartridge.

FEATURES	BENEFITS
Aerodynamic Media Stabilization	Highest Performance Reliability and Areal Density of any Removable Disk Drive
Unique Equalization Circuits	100 % Interchangeability
Design Simplicity	
Flexible Media	Lowest Cost 10 MByte Cartridge More Resistant to shock and Highly Resistant to Contamination
No Purge Cycle	Fastest Stop/Start of any High Performance Drive
On-Board LSI Controller	Only Disk Subsystem to Conform to Disk Standard (Size and Mounting)
SCSI Interface	Compatible with SCSI H/W and Protocol

Figure 25b. Features and Benefits of Bernoulli Disk Drives.

many floppy disks. They are also tough. Iomega representatives have thrown Bernoulli cartridges, which cost $80.00 each, around like frisbees to demonstrate this point. This disk is hard to beat when looking for a small mass storage system or for additional storage with a good backup facility.

VIII. TECHNOLOGY COMPARISONS

The conventional magnetic recording is in a state of renaissance at this time, with gains in density foreseeable for at least another decade. For the last 35 years, the data storage technology has been dominated by the conventional magnetic recording, and the rate of progress in the areal density, the key measure of merit, has continued undiminished, doubling about every 30 months for the last 30 years. The parameter which has made this possible has been the head-to-medium spacing, which has been reduced over the years from 25 microns to .3 microns, with current laboratory investigations now at .1 microns. Also important has been the precision mechanical employment, such as closed-loop servo systems, as well as improvements of media utilized.

This continuing resurgence of the conventional magnetic recording places increasing pressure on the optical technology. The pace of optical storage devices entering the marketplace has been rapidly increasing. A significant development was the introduction of DAD, and CD-ROM. These products will make the market acceptance of optical read-only, write-once, and erasable storage devices, easier. Although storage devices with the capacity and removability of optical media will clearly be needed in the future, conventional recording will maintain its dominance while other technologies will continue to flourish. In the next decade, it seems that optical technologies may be used as a complement to conventional recording. Although optical technologies have many advantages over the conventional recording, it is still an evolutionary technology which may take time for user acceptance and mass production. The conventional recording, on the other hand, is an established and understood technology which continues to meet users requirements.

Figures 26, 26a, 26b, 26c, and 26d depict the technology comparisons of the conventional and optical recording in terms of:

(1) density and data rate,

(2) access, capacity, and seek time,

(3) removability and cost of media,

(4) sturdiness and archivability, and

(5) head-disk gap and track servo.

	MAGNETIC		OPTICAL	
	3380	ULTIMATE	MODERATE (GaAs LASER, 0.5 NA LENS)	ULTIMATE (BLUE LASER 0.85 NA LENS)
LINEAR(1) DENSITY	~ 15K bPI	>40KbPI	~ 35K bPI	>80K bPI
TRACK DENSITY	>800 TPI	>3K TPI	~ 16K TPI	>10K TPI
AREAL DENSITY	12Mb/in^2	>120Mb/in^2	560Mb/in^2	>3200Mb/in^2
DATA RATE	3MB/s	~ 12 MB/s	3MB/s(2)	6MB/s(2)

(1) ASSUMING (2,7) CODE AT 1.5 BITS/TRANSITION

(2) X N FOR N ELEMENT ARRAY, PARALLEL RECORDING

Figure 26. Density and Data-Rate Comparison.

	MAGNETIC DISK	OPTICAL DISK
Direct Access	Fast	Medium
Sequential Access	Medium	Fast
Capacity	Large	Very Large
Seek Time	16 ms	85-500 ms

Figure 26a. Access, Capacity and Seek-Time Comparison.

	MAGNETIC DISK	OPTICAL DISK
Removability	No	Yes
Cost	Medium	Low

Figure 26b. Removability and Cost of Media.

	MAGNETIC DISK	OPTICAL DISK
Shelf Life	> 10 Yrs	> 10 Yrs
Encapsulation	No	Yes

Figure 26c. Studiness and Archivability.

	MAGNETIC DISK	OPTICAL DISK
Head-Disk Gap	.1 um	1000 um
Track Servo	Imbedded	Substrate Surface

Figure 26d. Head-Disk Gap and Track Servo.

IX. AN INTRODUCTION TO MODERN SOFTWARE

Technological advances in the storage and distribution of information have radically transformed the manner in which decisions are made today. These technologies have simultaneously expanded the variety and volume of information involved in decision making and have accelerated the pace at which decisions have to be made.

Unfortunately, the phenomenal growth in technologies that generate and distribute information has not been matched by a commensurate growth in technologies to monitor, filter, and analyze huge volumes of information. A technological imbalance has been created between the technical storage media and the technologies needed to make effective use of the stored information. The result has been an inability to exploit the full value of information gathered.

In this age of increased attention to the problems of information processing and utilization, one seeks for the formatted databases advances in techniques for data abstractions models, structures, accesses, retrievals, compressions and models, as well as differential files.

Due to the unformatted nature of textual databases, one seeks different advances in natural text-processing for capturing, storing, and retrieving large volumes of textual data. Machine text searching, automatic abstracting, and automatic indexing of full text documents, and selective dissemination of information are are now within the reach of the user. Both the formatted database and text database techniques can be use effectively to wade through the glut of the available information and support the decision making process of managers and users. These advances in techniques can offer a great return on investment, due to the number of programs and algorithms available in the public domain, many of which were prompted by the intelligence community. New advances are still being made in the research, academic and market places.

X. DATA ABSTRACTION TECHNIQUES

An abstraction hides the details of a set of algorithms and data and allows general and common properties of the set of algorithms and data to reveal. Thus, the abstraction is one of the main ways of structuring and visualizing vast amount of data and very complex algorithms. It is used to obtain categories of algorithms and data and to combine categories into more general categories. It has been used extensively in computer science to reduce complexity and aid understanding of algorithms and data.

An elementary form of abstraction distinguishes between the **token level** and the **type level**. A **token** is an actual value or a particular instance of an object. Abstraction is used to define a **type** from a class of similar tokens.

In terms of database objects, the abstraction is used in two ways: generalization and aggregation [Ref. 31].

In generalization, a set of similar tokens or a set of like types is viewed as one generic type. The token-type generalization is usually differentiated from the type-type generalization. The former process is referred to as "classification", while the latter process is called "generalization". For instance, viewing a set of individual employees as one generic type, employee, is considered classification, while viewing the types of employees and students as one generic type, person, is considered generalization.

The classification of tokens enhances understanding by allowing individual tokens to be grouped into types. Types can be further generalized into other, more general types. By using classification and generalization, the emphasis is placed on the similarities of objects and types while abstracting away their differences and details. Figure 27 illustrates this distinction.

An aggregation is the abstraction by which an object is characterized by its constituent objects. For instance, a person can be characterized by his name, address, and age. The aggregation can be use either at the token level or at the type level. For instance, the type employee can be characterized by the types: name, age, and address. An aggregation at the type level portrays a set of aggregations at the token level of the constituent types. Figure 28 illustrates this point.

Abstractions have been used informally in data management for a long time. While the aggregation is used during the file design to group fields of different data types in a common file, the generalization is used by introducing the notion of a file as a generic record type representing the properties of many records. Moreover, the record type has semantics and is no longer an uninterpreted set of records.

The abstraction can be used both with the bottom-up approach or the top-down approach. Using the bottom-up approach, an abstraction can be viewed as a synthesis of simple objects that enables one to understand a complex object. Starting with observed data, i.e., the tokens, to which one applies the classification to produce types, then the generalization and aggregation can be used to group and structure types into new generic and aggregate types.

Alternately, the top-down approach may be used to decompose complex types. Starting with a complex type, it can be decomposed into its components, through specialization, which is the opposite process to generalization, and instantiation, which is the opposite process to aggregation, to the token level. Typically, the bottom-up approach is used to understand a complex phenomenon and the top-down approach is used to design a complex object. Both methods can also be used together.

These two abstraction techniques are generally present in most data models. Some data models first define the tokens of information and then give structuring principles to combine and categorize them, while other data models enable the user to specify complex types which are associated with constituent types and eventually with tokens

Generalization is the Abstraction Technique by which a Group of Objects are Generically Classified

Example :

employee = generic [E1, E2, E3]

E1 = employee # 1

E2 = employee # 2

E3 = employee # 3

Figure 27. Generalization

Aggregation is the Abstraction Technique by which an Object is Constructed from its Constituent Objects

EXAMPLE :

employee = aggregate [Nm, E#, Ag, Ad]

Nm = name

E# = employee no.

Ag = Age

Ad = address

Figure 28. Aggregation

of information. Abstractions are used to give meaning to sets of objects, whether they are tokens or types.

A. CURRENT ABSTRACTION APPROACHES FOR STATISTICAL ABSTRACTS

The current approaches for statistical abstracts are **sampling** (aggregation) or **antisampling** (generalization). Figure 29 illustrates the difference between these two approaches. Sampling is the selection of constituent objects from the whole for an analysis and estimation of the nature of the whole. Antisampling is defined as the selection of a generic super set of the set in question in order to analyze the nature of the set.

In evaluating the relative merits of sampling and antisampling, there are numerous serious disadvantages to attempting to estimate statistics on a population by statistics on a random sample of that population. There are six disadvantages [Ref. 32].

Firstly, sometimes, the data have been aggregated in means, counts, and so on, as there have been large amounts of data from instrument readings in laboratory experiments. For example, much of the published U.S. Census data are in statistical forms to provide privacy protection for an individual's data values. Sampling aggregated data can be very tricky, and may not be possible without detailed information about the data before they are aggregated.

Secondly, sampling is inefficient in a paging environment. Assuming that the sample items are randomly distributed across pages, in exactly the same way that a formula is used for any set randomly distributed across pages, experiments [Ref. 33] have shown that sampling is going to be approximately the number of pages retrieved times less page efficient than a full retrieval of the entire database.

Thirdly, random sampling is also inefficient even when indices are used. Depending on how the index is stored, this may require more temporary storage space for the pointers to all the items in the set, and many index page accesses.

Fourthly, sampling is a poor way to estimate extremum statistics such as maximum, mode frequency, and bounds on distributional fits. Extrema have important applications in identifying exceptional or problematic behavior. Similarly, it is very poor for obtaining absolute bounds on statistics, which are important for many computer algorithms based on those statistics.

Fifthly, sampling is restricted to the nature of the sample itself. Given a sample, it is hard to speculate about properties of a subset, superset, or sibling of that set.

Lastly, a sample does not have semantics. It is of interest only as a sample and not as a set created by set intersections might be. As an alternative to sampling, the

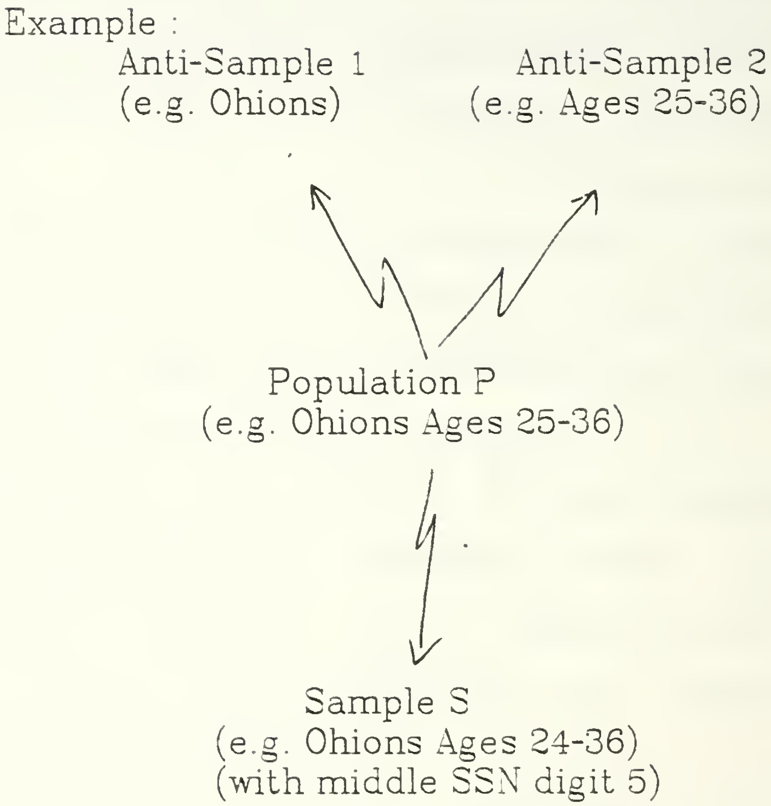

Example :
Anti-Sample 1
(e.g. Ohions)
Anti-Sample 2
(e.g. Ages 25-36)
Population P
(e.g. Ohions Ages 25-36)
Sample S
(e.g. Ohions Ages 24-36)
(with middle SSN digit 5)

Figure 29. Sampling vs. Antisampling

author in [Ref. 34] suggests the creation of a much smaller database, called a **database abstract**, which is a collection of simple statistics, such as means, maxima, etc., on important and frequently asked data in the database. The database abstract preserves most of the statistical content of the original data, in order to compensate for the above disadvantages. This database abstract is the major element in antisampling.

In utilizing this abstract, the processing speed can be traded off for storage. Since statistical databases often have much redundancy in attribute values and since these statistics can be predicted by other attribute values and statistics, they can be computed by programs on cheap processors, instead of expensive placing on secondary storage. A number of "reasonable-guess" rules can be used to infer statistical characteristics of the original data from the abstract. This technique provides an estimate for data in the initial stages of statistical analysis, emphasizing quick and rough estimates and visual displays. It directed towards hypothesis generation, not hypothesis testing.

This approach, by the employment of a database abstract of precomputed statistics plus inference rules, overcomes each of the abovementioned points and indicates as follows [Ref. 32]:

(1) The database abstract is an aggregation.

(2) Once set up, the database need not be paged at all. Paging of the abstract is low, since it is much smaller than the full database. Also, there are usually many sets of statistics relevant to a query, hence fewer retrievals are necessary than the retrievals for the same query without the statistics on the full database.

(3) Database index pages are used efficiently for the same reasons.

(4) Antisampling handles extremum statistics well since it can use extremum statistics of the entire database as bounds.

(5) Many rules explicitly address such cases as extensions of a set to supersets and restrictions of a set to subsets.

(6) Sets in the database abstract have an explicit semantics.

This approach provides a new alternative to sampling for exploring a large data population at low cost.

B. AN OVERVIEW OF THE ANTISAMPLING APPROACH

This top-down approach to low-cost estimations of statistics on a large computer database consists of a precomputed set of statistics known as a database abstract and a set of inference rules. This new approach starts with a user and a database. The database is preprocessed to create a database abstract, which is a collection of simple statistics (the mean, maximum, mode frequency, etc.) on important and frequently asked-about sets in the database. The user interacts with an interface to the database abstract, and asks the same statistical questions that he would ask the full database, if he had more time or space. If an answer is not in the database abstract, an estimate and bounds on the estimate are inferred for the answer from rules.

There are four dimensions to this rule taxonomy [Ref. 34]. These are:

(1) The statistical dimension, such as means and maxima.

(2) The characteristic dimension, such as exact answers, bounds and estimates.

(3) The computational dimension, for example, what forms of queries.

(4) The derivation dimension, for example, from where the results derived. For an example of the rule taxonomy, see Figure 30.

Given the disadvantages of the sampling approach it would appear that antisampling approach opens up a broad area for future research. Antisampling is not just another sampling method, but something fundamentally different, and subject to quite different advantages and disadvantages than sampling. Although some of its advantages have been discussed, one disadvantage that has not been mentioned is the amount of details that remains to be worked out, such as to increase the number of rules and to get better estimates.

Some new directions for further applications have been outlined, as well as extensions of this technique [Ref. 34]. Some extensions include rules for correlations, causations, rules for intensional knowledge, rules for prototypes,

Rule : Largest Item in the Intersection
of the Two Sets cannot be Larger
than the Minima of the Maxima
of the Two Sets

Statistical Dimension: Rule for Max Statistic

Characteristic Dimension: Upper Bound

Computational Dimension: Intersection of the Sets

Derivation Dimension: Basic Mathematics

Figure 30. Example of the Rule Taxonomy

dependencies, quantifiers, and nulls. More sophisticated control structure can be readily implemented. Special-purpose hardware could improve performance of the system. For example, obtaining the database abstract is computationally expensive, and special devices for computing the basic aggregate statistics on the database might be very helpful, perhaps as components in disk drives. Such devices would also improve answer speed for arbitrary statistical queries on the database. Moreover, since paging is a major cost in this system, the use of the read-only memory for the database abstract might significantly improve its performance. This could be quite cost effective for much used databases like the U.S. Census.

The idea of providing for the first time an alternative to sampling, for estimating characteristics of a large data population at low cost, is very appealing.

C. CURRENT ABSTRACTION METHODS FOR CONTENT ABSTRACTS

The current abstraction methods for content abstracts utilize *content analysis concepts* [Ref. 35]. They are *symbol-matching*, *content-oriented*, *element retrieval* and *information retrieval* methods.

The content analysis is a process of delineating what a sentence says. Hence, only humans can interpret and fully comprehend the meaning of a natural language sentence that may be incomplete, idiomatic and valid only in the context of the dialogical communication. Moreover, only humans can appreciate the differences in the wording of sentences, or can easily interpret the jargon.

In using this technique, the body of text that is to become the retrieval database consists of sentences. A sentence means an English sentence or partial sentence with no formal restrictions on its structure or organization. Sentences are the foundation unit of which the database is composed because they are the basic unit of the human communication.

Using the symbol-matching retrieval technique, the database is searched to locate the data element that contains a certain symbol or a sequence of symbols. Once this symbol is located, the data element may be retrieved as a whole or may be subject to manipulation. An example of such a technique is the keyword-indexing technique in which the symbol being searched for is an English word. However, this technique is not considered very viable, since the data cannot be counted on to contain appropriate key symbols for indexing, as in the case with the example, "Hit the deck".

With content-oriented retrieval, a data element is identified according to its content or meaning, rather than by the symbols or keywords. Too often, natural English sentences may not be complete, such that a pronoun is used instead of a noun, which may be implied. For example, "He was not able to continue", is a sentence that has no key symbols, and hence only content-oriented retrieval would be appropriate. Moreover, either the meaning of sentences is conveyed by the style; or the composition is frequently more important than the meaning of individual component elements. Considering the sentence "Time flies", the meaning is entirely different when each word is considered separately rather than jointly. The preferred approach is whenever a content-oriented technique is used, a symbol- matching technique is also used in conjunction with it.

The content abstraction encompasses the following two types of retrieval: element and information retrieval. The element retrieval returns the entire data element satisfying the query, whereas information retrieval returns only the answer extracted from the data element. For example, giving a database that includes the following sentence: "Company X will conduct a reconnaissance at 0600 hours", and the considering the query, "what is Company X doing at 0600 hours?" An element retrieval system would retrieve the entire sentence, while an information retrieval system would only reply with "reconnaissance". Either retrieval technique is acceptable, depending on the output desired. The contents analysis technique is performed on each sentence by filling in a standard abstract form (see Figure 31). The technique begins by assigning an identification number to each sentence. Then, the content of each sentence is analyzed; all applicable properties are checked. These five properties are [Ref. 35]:

(1) Type - whether it is declarative, interrogative, imperative, responsive, exclamatory, and acknowledgment. Type is applicable to every sentence.

(2) Nature - whether it is a requisition, conclusion, characteristic, valuation, or recommendation. Nature is not applicable to every sentence.

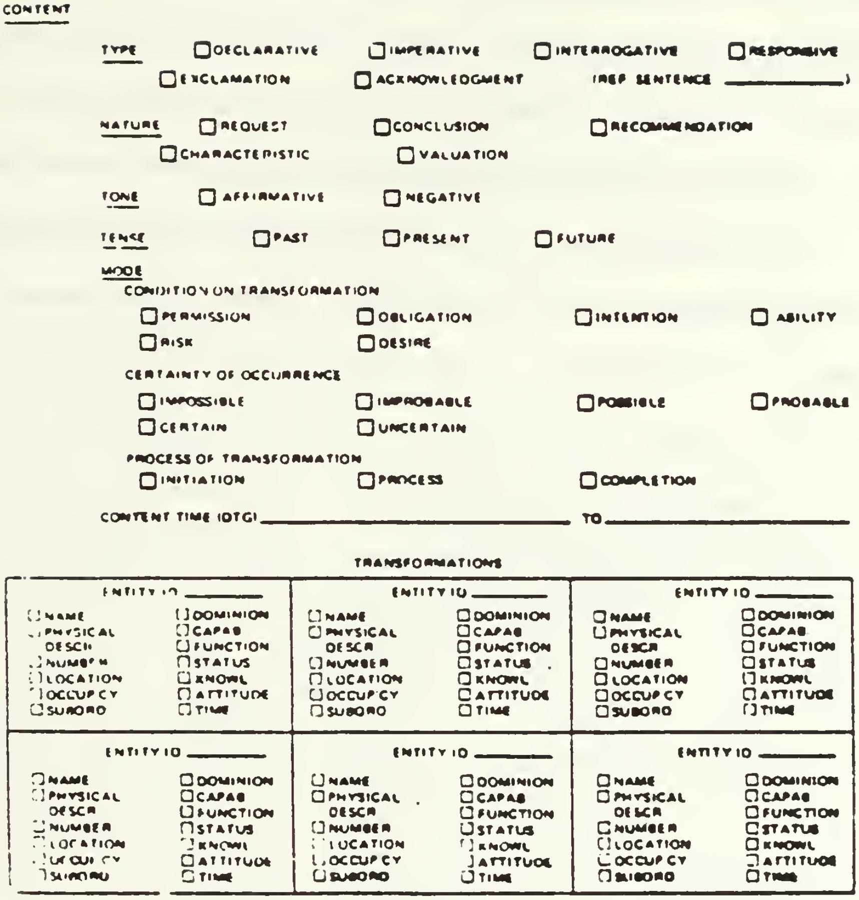
(3) Tone - whether it is affirmative, uncertain, or negative. Tone is applicable to every sentence.

(4) Tense - whether it is past, present, or future. A tense is applicable to every sentence.

(5) Mode - whether it is altering the characteristics of the things that the sentence talks about, such as obligation, intention, permission, ability, risk, and desire; occurring with certain probability and having a finite duration such as at the beginning, somewhere in process, or terminating. Mode is not applicable to every sentence.

Mode alterations, also known as transformations, are identified by denoting every entity mentioned or implied in the particular sentence with its identification number entered in the appropriate blank. All attributes (characteristics) applying to that entity are checked.

Although this abstraction technique appears to cover the broad range of meaning that can be contained in an English sentence, there are, however, numerous constraints with this approach. If an extremely detailed abstraction scheme is utilized, that accurately



CONTENT

TYPE ☐ DECLARATIVE ☐ IMPERATIVE ☐ INTERROGATIVE ☐ RESPONSIVE
☐ EXCLAMATION ☐ ACKNOWLEDGMENT (REF SENTENCE ____________)

NATURE ☐ REQUEST ☐ CONCLUSION ☐ RECOMMENDATION
☐ CHARACTERISTIC ☐ VALUATION

TONE ☐ AFFIRMATIVE ☐ NEGATIVE

TENSE ☐ PAST ☐ PRESENT ☐ FUTURE

MODE
CONDITION ON TRANSFORMATION
☐ PERMISSION ☐ OBLIGATION ☐ INTENTION ☐ ABILITY
☐ RISK ☐ DESIRE

CERTAINTY OF OCCURRENCE
☐ IMPOSSIBLE ☐ IMPROBABLE ☐ POSSIBLE ☐ PROBABLE
☐ CERTAIN ☐ UNCERTAIN

PROCESS OF TRANSFORMATION
☐ INITIATION ☐ PROCESS ☐ COMPLETION

CONTENT TIME (DTG) ____________________ TO ____________________

TRANSFORMATIONS

ENTITY ID ________ ☐ NAME ☐ DOMINION ☐ PHYSICAL DESCR ☐ CAPAB ☐ FUNCTION ☐ NUMBER ☐ STATUS ☐ LOCATION ☐ KNOWL ☐ OCCUPCY ☐ ATTITUDE ☐ SUBORD ☐ TIME	ENTITY ID ________ ☐ NAME ☐ DOMINION ☐ PHYSICAL DESCR ☐ CAPAB ☐ FUNCTION ☐ NUMBER ☐ STATUS ☐ LOCATION ☐ KNOWL ☐ OCCUPCY ☐ ATTITUDE ☐ SUBORD ☐ TIME	ENTITY ID ________ ☐ NAME ☐ DOMINION ☐ PHYSICAL DESCR ☐ CAPAB ☐ FUNCTION ☐ NUMBER ☐ STATUS ☐ LOCATION ☐ KNOWL ☐ OCCUPCY ☐ ATTITUDE ☐ SUBORD ☐ TIME
ENTITY ID ________ ☐ NAME ☐ DOMINION ☐ PHYSICAL DESCR ☐ CAPAB ☐ FUNCTION ☐ NUMBER ☐ STATUS ☐ LOCATION ☐ KNOWL ☐ OCCUPCY ☐ ATTITUDE ☐ SUBORD ☐ TIME	ENTITY ID ________ ☐ NAME ☐ DOMINION ☐ PHYSICAL DESCR ☐ CAPAB ☐ FUNCTION ☐ NUMBER ☐ STATUS ☐ LOCATION ☐ KNOWL ☐ OCCUPCY ☐ ATTITUDE ☐ SUBORD ☐ TIME	ENTITY ID ________ ☐ NAME ☐ DOMINION ☐ PHYSICAL DESCR ☐ CAPAB ☐ FUNCTION ☐ NUMBER ☐ STATUS ☐ LOCATION ☐ KNOWL ☐ OCCUPCY ☐ ATTITUDE ☐ SUBORD ☐ TIME

Figure 31. Standard Abstract Form

reflected all or most of the subtlety and complexity of the natural English sentences, an extensive training of the great intelligence for the analyst would be required. If, on the other hand, a sentence analysis scheme simple enough for the low-level personnel is used, it could be too inaccurate. Moreover, this schema is difficult to deal with sentences that have word usage errors. Also, semantic interpretation depends on how good the input analysis is, and contents abstracts do require human intervention.

This technique can be enhanced by extending some of its capabilities. First of all, the abstraction form can be readily automated. Secondly, an inferential retrieval technique can be utilized when an answer to a query is not directly contained in the database. For example, if a database contained the sentences, "Company X is a part of Battalion Y", and "Battalion Y is on maneuvers in Honduras", and the query has been, "Where is Company X ?", a sophisticated element retrieval system should be able to return the two sentences concerning Company X and Battalion Y, thereby enabling the inference that Company X is probably in Honduras. Thirdly, the whole process can be automated, thereby saving time, cost, and effort.

D. THE AUTOMATIC DATA ABSTRACTING

Despite recent advances in technology, high quality abstracts must still be produced manually. Consequently, the preparation of abstracts and their associated indices accounts for over half of the cost and time required for publication. Thus one must take into account the factors of production time and cost when comparing the manual with the mechanized abstracting methods.

Two alternative solutions exist to overcome the problems associated with the manual abstract production:

(1) Use author-prepared abstracts as a prerequisite to publication.

(2) Mechanize the abstract production.

Abstracts from authors, although editors and publishers have in recent years made an effort to get good abstracts from their authors, are of questionable value. Thus, together with the increasing shortage of qualified abstractors, the factors of time, cost, and value have lent impetus to a trend toward the automatic generation of abstracts and indices. This trend has caused increased emphasis to be placed on the abstract as the locus of data for automatic retrieval systems. This, of course, necessitates the creation of high-quality abstracts.

1. System Requirements

Some of the basic requirements which an automatic abstracting system must fulfill, include the unit of data to be processed, methods of sentence selection, notions of contextual inference, intersentence reference and coherence criteria [Ref. 36].

a. The Basic Unit of Data

A language processing program, for efficiency, should consider the largest independent item in its database as its basic unit. Thus, in automatic abstracting the basic unit is the original article. It would be inadequate to consider any approach in which either paragraphs or sentences are considered basic units, because of the interdependence between these elements and the remainder of an article. A program which operates on interdependent units bears the burden of carrying data from one unit to the next. An automatic language processing program must also be able to identify and manipulate the elements of its basic data unit, whether these elements be words, phrases, clauses, or sentences.

b. Sentence-Selection Methods

In order to develop criteria for selecting sentences to form an abstract, it is necessary to analyze the conditions under which various methods of sentence selection are successful. It is apparent, however, that an abstract can also be produced by rejecting sentences of the original which are irrelevant to the abstract. Therefore, it is no wonder that methods of rejecting sentences also deserve intensive analysis.

c. Notions of Contextual Inference

Contextual inference is the basic concept underlying sentence selection or rejection. Thus, given a data element (word or word string) within a sentence and some surrounding context, using contextual inference, it is generally possible to infer whether a sentence should be rejected or selected for inclusion in an abstract. Contextual inferences may be made based on either the physical arrangement of the elements of a document, called the *location* method, or on word strings which comprise these elements, called the *cue* method. These two basic approaches to the making of contextual inferences are discussed as follows:

(1) *The Location Method.* The location method is based on the physical arrangement of the elements of an article. This arrangement can be described in terms of the location of a sentence with respect to how long the document containing the sentence is, or in terms of the location of phrases, clauses or words with respect to how long the sentence, containing these elements is.

The first location type, called sentence location, is governed by the style of the author or the editor, with general writing guides providing advice about the placement of sentences within an article. Since it is not possible to dictate the matter of the style, the location of a sentence does not convey an unequivocal criterion for sentence selection or rejection.

The second location type is actually a sentence description, since the location of phrases and words within the sentence is subject to grammatical rules to which authors and editors adhere.

Since a sentence is a string of words terminated by a period, question mark, or semicolon, the punctuation, hence, plays an important role in the location method. Each punctuation mark serves a specific purpose.

Both the question mark and the semicolon have a rather unambiguous use; the period, however, is used in abbreviations, and numbers, as well as at the end of a sentence. These different usages must be differentiated in order to properly analyze sentences.

Commas, like periods, can also have several uses. They can separate items in series, parenthetical expressions, and clauses, as well as occurring in numbers. Serial or numerical commas do not really provide sufficient information to make the determination of either rejection or acceptance. Parenthetical commas, on the other hand, since they normally merely elucidate, can be rejected, because they can be removed without affecting the meaning of the sentence. Commas that separate clauses are more important in this rejection/acceptance scheme, since they delimit the leading clause from clauses which qualify it. Second or subsequent clauses generally modify the first clause, thereby concluding that the first sentence is essential to the meaning of the sentence. However, second and subsequent clauses can be rejected, and still obtain a sensible result. For example, the sentence, " Company X is the best fighting unit in the Division, because its commander is Italian", is just as grammatically correct and coherent, if its subordinate clause are removed (rejected), with the result, "Company X is the best fighting unit in the Division". Of course, this rejection depends entirely on the abstract desired.

When dealing with clauses, further reductions can be made by removing prepositional phrases. Thus, the sentence, "Company X is the best fighting unit", perfectly expresses the original meaning, depending, of course, on the abstract desired.

Thus, the location method, which is based on the general hypothesis that certain headings precede important passages and that topic sentences occur early or late in a paragraph, is based on contextual inference. Punctuation, words, phrases and clauses are sentence elements whose context can be used to infer whether a sentence has value for an abstract or not [Ref. 36].

(2) **The Cue Method.** The method of a sentence selection or rejection based on **cue words** is called the cue method. Cue words are words that provide unambiguous clues to such things as opinion and subjectivity, as well as to some positive notions. These cue words are normally contained in the dictionary, along with codes, which indicate the frequency of occurrence within a context (see Figure 32).

The Cue method provides a powerful approach to sentence selection or rejection. The method depends on the fact that it is possible to decide what should or should not be included in an abstract, based upon the presence in the original article of particular words or combinations of words. For example, words that may indicate the purpose of a document, such as, "My thesis", is an excellent candidate for acceptance. Opinions and subjective notions, which should not be included in an abstract, can be identified by such cue words as "obvious, or believe". Moreover, the code of a cue word may depend on its position in a sentence. A sentence starting with "A" or "Some" is more likely to present detailed descriptions than a sentence which contains either of these words in a more central location of the sentence. The reason for this is that these words have a strong quantitative function when they appear at the beginning of a sentence. Similarly, sentences which

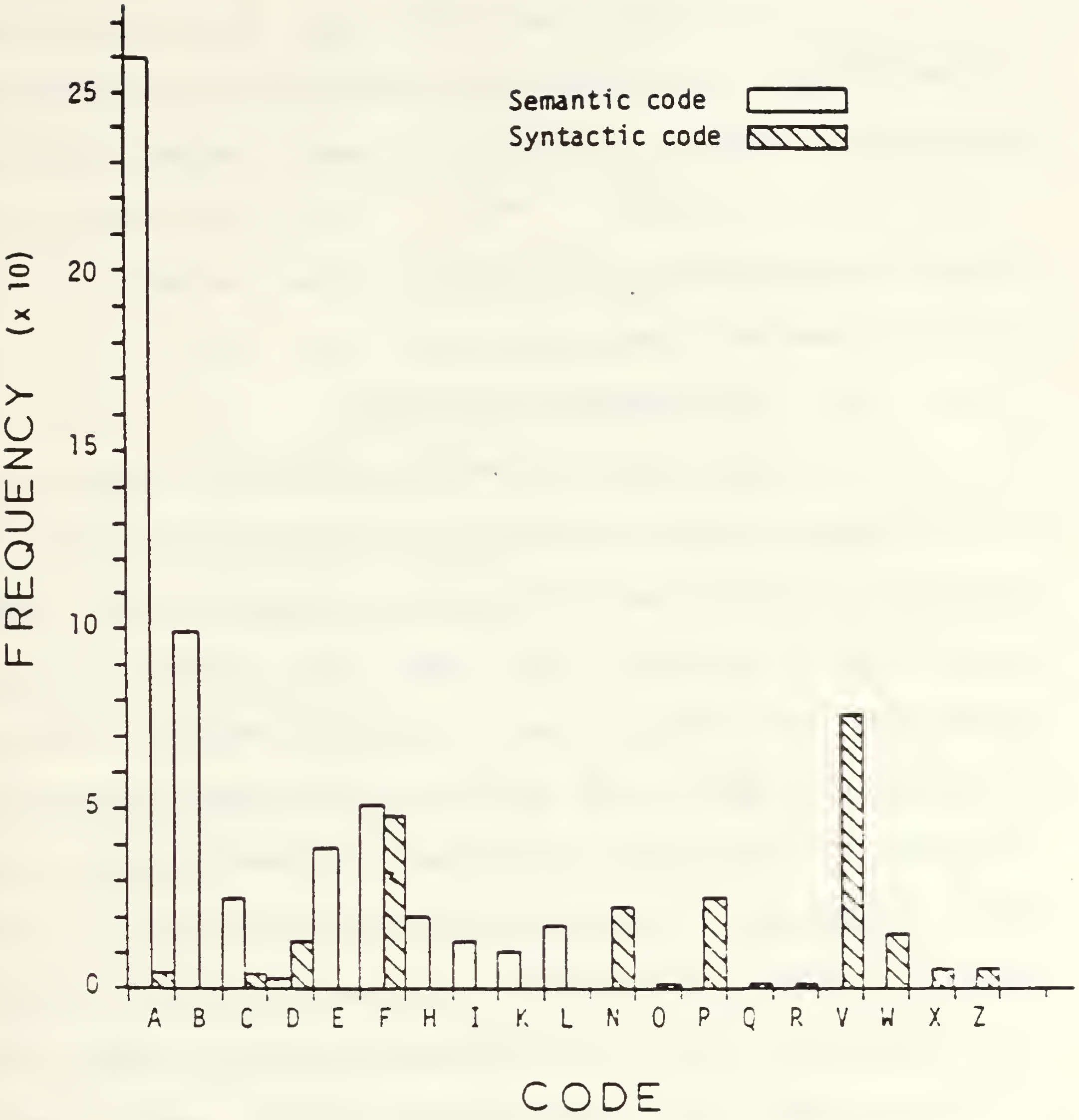


Figure 32. WCL Entries vs. Semantic Weight

begin with participles are usually conditional in nature, indicating assumptions or conjectures.

Also cue words may be used to identify parenthetical expressions, idiomatic expressions and cliches as well as to carry syntactic roles in cases where there is no ambiguity. Thus do cue words make it easier to determine whether a sentence or phrase should be selected for the abstract [Ref. 36].

d. Intersentence Reference

Intersentence references give quite a bit of information about the logical relationships within the text material, by the use of multiple clauses, cue words, and title words. When more than one clause exists in a sentence, the first clause is indispensable to the meaning of the sentence, and generally the first clause will also contain intersentence references if there is any. Words in the second and subsequent clauses which require antecedents usually refer to the first clause. Some cue words that indicate an intersentence references are " these, they, and it". Now, when these words have multiple uses, additional criteria are then required in order to determine if there is an intersentence reference. Such criteria includes discovering patterns in the use of words such as "it" to make possible the use of these words to detect intersentence references. The following logical rule could emerge: "It" in the first or only clause indicates intersentence reference unless followed closely by "that". For example, the sentence, "It is known that Saint Nick represents Santa Clause", does not refer to a previous sentence, while the sentence, "It was still rolling", does. Of course, there will always be exceptions to this rule.

There are, however, intersentence references that do not make use of any cue words; instead, they use the name of the antecedent rather than a pronoun. Intersentence references of this type can be detected by the title words. If any words, which are defined as non-function words such as those words which are not articles, conjunctions, prepositions etc., occur in adjacent sentences, the sentences are likely to be closely related. An example would be the sentences, "Hydrogen and oxygen form water" and "Water is colorless".

This method of selecting sentences by title words is a special type of intersentence references that is between the title and the rest of the sentences in a document. The title method has the premise that the author generally describes in as few words as possible the essence of his paper. Thus, it can be safely assumed that the words of the title are well chosen and of high significance [Ref. 36].

2. The Coherence Consideration

In determining sentence selection or rejection, the coherence of the abstract must be considered. For example, if there is a sentence that required an antecedent to be included in an abstract, it would be necessary to determine if the previous sentence has been removed and if necessary to reinstate it. If the restored sentence also requires an antecedent, the procedure must be repeated. But, if several sentences would have to be reinstated because of the required antecedents of one sentence, then that sentence might as well be rejected.

3. The System Configuration

The automatic abstracting system consists basically of a dictionary, called the Word Control List (WCL), and a set of rules for implementing certain functions specified for each WCL entry. To automatically produce abstracts, it is necessary to identify and eliminate certain sentences of the document. It is also necessary to identify and select a few sentences for the abstract, and to retain, by default, as well, certain sentences for the abstract (see Figure 33). These three methods of sentence handling are discussed below.

a. The Sentencial Elimination

The exclusion of sentences from the abstract involves the detection of words or strings of words which identify sentences giving historical data, results of previous work, examples, explanations, speculative material and so on. A set of word strings needed is in the order of a few hundred in order to eliminate up to 90% of the sentences of a document. Such word strings are incorporated in WCL.

WCL consists of an alphabetically ordered set of words and phrases, which are referred to collectively as **word strings**, and one or two associated codes (see Figure 33). The entries in WCL are treated as functions and each has two arguments: a semantic weight (see Figure 33a) and a syntactic value (see Figure 33b). Each function returns a value which indicates whether the sentence is a candidate for retention or deletion. Entries in WCL may be varied as desired without necessitating any changes in the programs of the system. In general a WCL entry can be represented as: Word String multiplied by Weight multiplied by Value.

b. The Sentential Retention

In general, the semantic weight of a WCL entry can be either positive or negative. WCL entries, which have positive semantic weights, are retained; otherwise, they are rejected. Such word strings in a document as, "this paper", "this study", or "present work", are retained. It is nearly certain that the author of such a document is about to say what the document is all about, and should, therefore, be retained, because it does belong to the abstract.

In addition, sentences which contain personal pronouns such as "we", "I", or "our", are good candidates for retention. Finally, sentences which contain significant title words, such as words of the title which are not in WCL, and which do not contain word strings having strongly negative semantic weights, should also be retained. There are no other instances in which a sentence is deliberately retained, although a small number of sentences of the document belonging to the abstract will be retained by default.

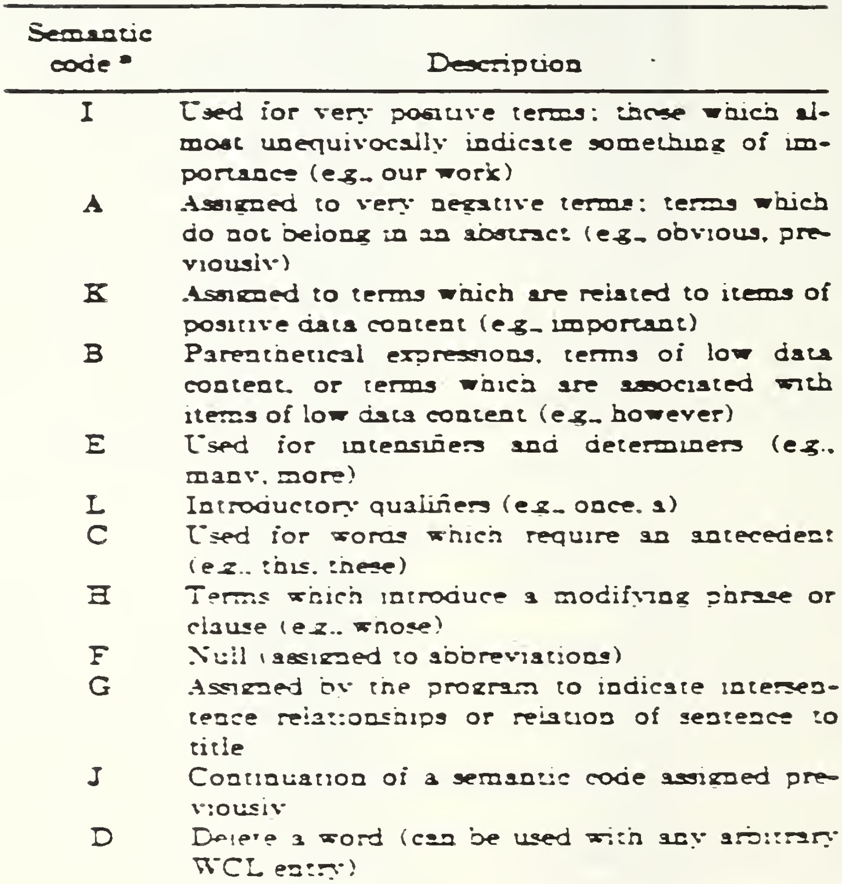
c. Rules for Implementing the Functions in WCL

A viable solution to determine whether a sentence of the document is a member of the abstract or not, using a two-valued membership criterion, is to impose an ordering on the semantic weights [23]. This ordering provides considerable flexibility by incorporating the rules in the program and by supplying the semantic weights externally. The rules can be altered without the necessity for changing WCL, and WCL can be altered independently of the rules. There are a total of 19 rules.

Since the implementation of the semantic weights requires some syntactic information, a partial

```
NAMELY*B
NEVERTHELESS*B
NEXT SECTION*A
NEXT SECTIONS*A
NO ACCURATE*A
NO ATTEMPT*B
NOR* *C
NOT ALWAYS*B
NOT BEEN*A
NOT CLEAR*B
NOT IMPORTANT*A
NOT*L
NOTED*A*V
NOTEWORTHY*K
NOW*A
O .*F*F
OBSCURE*A
OBVIOUS*A
OBVIOUSLY*A
OF ABOUT*H
OF COURSE*A
OF* *O
OFFER* *V
OFTEN*E
ON THE OTHER HAND*B
ON WHICH*B*P
ON* *P
ONCE*L
ONE CAN*A
ONE OF*B
ONE*E
ONLY*E
OPINION*A
OR*L*C
OTHER*E
OTHERS*E
OUR WORK*I
OUR*K*N
OVER* *P
OVERT*B
P .*F*F
PARAGRAPH*A
PARTICULAR*A
PAST*A
PERHAPS*A
PERMITTING*B
PLACE* *V
PLACED* *V
POINT*B
POINTED OUT*A
POSSIBILITIES*A
POSSIBILITY*A
POSSIBLY*A
POTENTIALITY*A
POTENTIALLY*A
PRECISELY*A
PRELIMINARY*A
PRESENT DAY*A
PRESENT PAPER*I
PRESENT YEAR*A
PRESENT*B*V
PRESENTED HERE*I
PRESENTED*A*V
PRESENTS* *V
PREVIOUS*A
PREVIOUSLY*A
PRIMARILY*B
PROBABLY*A
PROBLEMS*B
PROGRESS*A
PROGRESSING*E
PROPOSED*A
PROVIDE* *V
PUBLISHED*A*V
PURPOSE*K
Q .*F*F
QUESTION*A
QUESTIONABLE*A
QUESTIONED*A*V
QUITE*A
R .*F*F
RATHER*A
```

Figure 33. Word Control List



Semantic code[a]	Description
I	Used for very positive terms: those which almost unequivocally indicate something of importance (e.g., our work)
A	Assigned to very negative terms: terms which do not belong in an abstract (e.g., obvious, previously)
K	Assigned to terms which are related to items of positive data content (e.g., important)
B	Parenthetical expressions, terms of low data content, or terms which are associated with items of low data content (e.g., however)
E	Used for intensifiers and determiners (e.g., many, more)
L	Introductory qualifiers (e.g., once, a)
C	Used for words which require an antecedent (e.g., this, these)
H	Terms which introduce a modifying phrase or clause (e.g., whose)
F	Null (assigned to abbreviations)
G	Assigned by the program to indicate intersentence relationships or relation of sentence to title
J	Continuation of a semantic code assigned previously
D	Delete a word (can be used with any arbitrary WCL entry)

Figure 33a. Semantic Attributes For WCL Entries

Syntactic code	Description
A	Article
C	Conjunction
D	Delete the word
F	Null word
J	Continuation of a previous syntactic value
N	Pronoun
P	Preposition
O	Exclusively assigned to OF
Q	Exclusively assigned to TO
R	Exclusively assigned to AS
V	Verb
W	Auxiliary verb
X	Exclusively assigned to IS, ARE, WAS, and WERE
Z	Negatives

Figure 33b. Syntactic Values for WCL Entries

syntactic analysis of each sentence is performed. This analysis is carried out through the use of the syntactic values associated with entries in the WCL, in conjunction with procedures implemented within the program. One of ten possible syntactic values may be associated with an entry in WCL. These syntactic values distinguish auxiliary verbs within verbs and "is", "are", "was" and "were" are distinguished from other auxiliaries. Similarly, the preposition "to", "as" and "of" are distinguished from each other and from other prepositions [Ref. 37].

4. Data Structures for Automatic Abstracting

In implementing the data structures some features of lists, defined as pointers to the data, and some features of tables, defined as storage of word attributes, are incorporated. Data structures consists of the following three (see Figure 34):

(1) A work area, where the text is stored throughout the processing;

(2) An attribute vector, containing pointers to each word of the text, and the length, semantic and syntactic attributes for the corresponding words. Textual properties such as the capitalization could also be incorporated in the attribute vector. The nth element of the attribute vector corresponds to the nth word of the text in the work area;

(3) An alphabetic vector, which defines the alphabetic rank of the words of the text. The nth element of the alphabetic vector contains the number of the attribute-vector element which corresponds to the nth word

in alphabetic sequence. The alphabetic
vector permits matching against a dictionary
without reorganizing the data.

The attribute and alphabetic vectors are combined to form a table; it is possible to reuse the space of the alphabetic vector after all alphabetic processing has taken place, hence combining the vector results in an overall space savings.

5. Conclusions

The abstracts that have been obtained with this system have been of sufficiently good quality. Results have demonstrated that a 80% to 90 % reduction of text is obtained, and at costs comparable to the cost of those produced manually [Ref. 37]. Furthermore, one of the main advantage of this automatic system is that the processing programs are text-independent. Thus, abstracts written in any language with a similar structure to English could be easily obtained, simply by substituting appropriate WCL. Another advantage is that abstracts with a particular bias can be produced, simply by variation of the WCL. Thus, tailor-made abstracts can be produced.

Research into automatic abstracting continues to be attractive because of the following factors:

(1) Manual abstracting is expensive and time consuming;

(2) Machine readable journals will probably become more widely available in the near future due to the increasing use of computer controlled composition in the printing industry. This

CONTENTS OF WORK AREA

```
MACHINE ADDRESS:   0   4     10  14  18   23       31       39      46
                   '   '     '   '   '    '        '        '       '
TEXT:             The rocks did not have sharply angular corners.
```

TABLE CORRESPONDING TO THE WORK AREA

Implied vector element number	Attribute vector word length	 word address	 attributes		Alphabetic vector
0	3	0	-	A	6
1	5	4	-	-	7
2	3	10	-	W	2
3	3	14	L	Z	4
4	4	18	-	W	3
5	7	23	-	-	1
6	7	31	-	-	5
7	7	39	-	-	0
8	1	46	.	.	8
Byte	1	2-4	5	6	7-8

DATA STRUCTURE

Figure 34. Contents of The Work Area

will provide a relatively cheap and convenient database for experimentation;

(3) Given the availability of machine readable journals, automatic abstracting will be much cheaper and faster than manual abstracting;

(4) Automatic abstracting produces abstracts in the machine-readable form ready to be used for later processing steps;

(5) While automatic abstracts may never be as good as manual abstracts, they are good enough for practical purposes.

E. OTHER ABSTRACTING METHODS

Other specific experimental automatic abstracting systems include the four following systems.

The first is ADAM (The Automatic Document Abstracting Method), which has been designed to produce indicative abstracts, i.e., abstracts which enable the reader to judge whether or not one needs to read the original document. Most automatic abstracting methods differ from ADAM in two important respects: they rely heavily on statistical criteria as a basis for sentence selection and rejection, and are designed to select sentences for abstracts. In contrast, ADAM uses statistical data only peripherally and is designed for sentence rejection rather than selection. The results of this experiment are as follows:

(1) The quality of ADAM extracts, while lower than that of good manual abstracts, is functionally adequate;

(2) ADAM requires, on the average, 0.6 sec of computer time per document;

(3) ADAM needs a specialized WCL for each subject area;

(4) ADAM abstracts can be improved by simple manual editing [Ref. 38].

The second is ASISA (The Analysis of Semantic Information Structure in the Abstract), which is a method to extract significant phrases in the title and the abstract of scientific or technical document. The method is based upon a text structure analysis and uses a relatively small dictionary. The dictionary has been constructed based on the knowledge about concepts in the field of science or technology and some lexical knowledge, for significant phrases and their component items may be used in different meanings among the fields. A text analysis approach has been applied to select significant phrases as substantial and semantic information carriers of the contents of the abstract.

This system consists of five modules (see Figure 35):

(1) Text Input Module, which reads in a card size record at a time and decides whether it is retained or not according to both a sign of the first column of the record and its preceding record sign. The records of the title and abstract are concatenated with each other to form a character string as a whole. Then the character string is transferred to the next module. The records of the keywords set are resolved into a collection of keywords and stored in the memory to be compared with the extracted phrases, later. The other records are sent to the Output module without any processing;

(2) Term Extraction Module, in which the term as a candidate of the meaningful item is extracted from the character string of the title or every sentence of the abstract by dividing it with the delimiters. Thus, the term obtained by this process is recognized as a meaningful item and transferred to the next module;

(3) Term Checking Module, which consists of both looking up the dictionary and the ending processing. The dictionary has been constructed only for the purpose of extracting meaningful items bases upon the knowledge about concepts of the field.

(4) Phrase Generation Module, in which the terms accepted by this module have consequently been classified into one of the following four kinds: deletion words (D), adjective words (A), weak noun words (W), and strong noun words (N). Using these symbols, the character string can be viewed as a sequence consisting of the symbols corresponding to a sequence of words in the string in order.

(5) Output Module, which has two different kinds of outputs in the system, that is the output for every document and output for a set of documents.

Following are some of the results of this particular method. Significant phrases represented in the abstract have been effectively extracted and very compatible with the author prepared keywords. The number of whole noun phrases extracted from the abstract is on the average 1.5 times as many as the author prepared keywords, and the title is not an adequate source for semantic contents analysis of the document, for 80% of them consists of 1 to 2 words [Ref. 39].

The third is SIE (The Specialized Information Extraction), whose task it is to obtain information automatically from a natural language text, in which some of this text is of a highly stylized nature with a restricted semantic domain, and place it in the database. Specifically, SIE is designed to extract information regarding chemical reactions from experimental

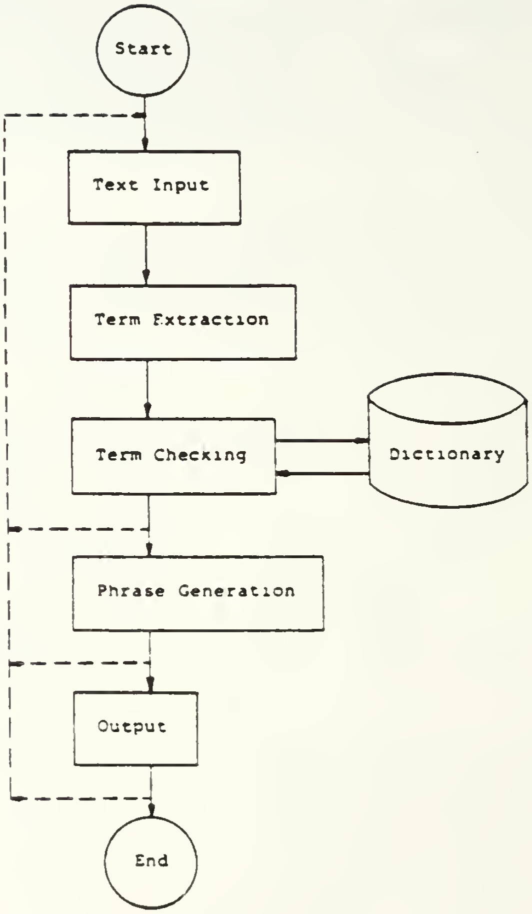

Start
Text Input
Term Extraction
Term Checking
Dictionary
Phrase Generation
Output
End

Figure 35. System Configuration

sections of papers in the chemical literature and to produce a data structure containing the relevant information.

In evaluating this system, the following three measures have been utilized:

(1) Robustness, which is the percentage of inputs handled;

(2) Accuracy, which is the percentage of those inputs handled which are correctly handled;

(3) Error rate, which is the percentage of erroneous entries within incorrectly handled input.

The results are as follows: the robustness was 92%, but the accuracy was only 78%, and since these were full of errors, no error rate is computed. The reason for this is that the most difficult aspect of SIE is the provisions of a safety factor, which is an ability for the system to recognize inputs that it cannot handle. It is clear that one can create a system that is robust and acceptably accurate which has unacceptable error rates for certain inputs. If the system is to be useful, it must be possible automatically to determine which documents contain unacceptable error rates. If the safety factor can be improved, SIE offer a promising area of application.

SIE programs are more feasible than automatic translation because the restrictions has lessen the ambiguity problems. This is true even in comparison to other tasks with a restricted subject matter, such as natural language computer programming or database query. Furthermore, these latter tasks require a very low error rate in order to be useful, because users will not tolerate either incorrect results or constant queries. While SIE programs would be successful if they could produce results in, say, 80% of cases, it is required that information extraction be done by humans in other abstracting methods. Even small rates of undetected errors would be tolerable in many situations, though one would naturally wish to minimize them [Ref. 40].

The fourth is MIF (The Model for Information Formatting), which is an approach that uses an explicit grammar of English within the domain to derive a tabular representation of the information in a message narrative. This method employs artificial intelligence techniques to extract information. In simplest terms, an information format is a large table, with one column for each type of information which can occur in a class of texts and one row for each sentence or clause in the text. The information carried by the narrative can be extracted much more easily from the format entries than from the original text. Mapping the text into an information entails four processes, which are discussed later.

The long term goal of this work is to develop capabilities that will enable systems to handle a broad spectrum of military messages, from highly formatted messages with little English description to messages consisting entirely of English narratives. Currently, the experimentation is limited to Navy operational reports.

The narrative portion of each message is automatically transformed into a series of format entries using a procedure which involves four stages of processing: parsing, syntactic regularization, mapping into the information format, and format normalization.

First, the text sentences are parsed using the broad-coverage string grammar, which has been extended to handle the sentence fragments which appear in these messages.

Second, the parse trees are syntactically regularized by a series of transformations in order to simplify the subsequent mapping into the information format (i.e., the various types of clauses such as passive, relative, sentence fragments and other and others are transformed into simple active assertions).

Third, the phrases in the syntactically regularized parse trees are moved, one by one, into the information format. This process is controlled in large part by the semantic word classes associated with each word. These classes, along with syntactic information about the word, is recorded in each word's dictionary entry.

The fourth stage, adds to the format certain information which is implicit in the text. This includes missing arguments, such as subject and object of verbs, and pronouns which can be reconstructed from earlier format entries. Results from this study has underlined the importance of continued research into two areas, knowledge representation and robustness, in order to extend this method into more diverse applications [Ref. 41].

Currently, the most promising application of all the above techniques is in the extraction of information from the highly restricted semantic domain of specialized technical journals.

XI. DATA ACCESS AND RETRIEVAL METHODS

An important task of an operating system is the maintenance of files. Certain facilities are provided by an operating system for creating, destroying, organizing, reading, writing, modifying, moving, copying, and controlling access to files. The component of an operating system that provides these facilities is usually referred to as a file system. One of the roles that this file system plays is an interface between a program and the files that the program expects to access. Another role of a file system is as supervisor that monitors files. The user communicates indirectly with a file system via an operating system through a set of predefined commands commonly called a job command language, assembly language programs, or programs written in a high-level programming language. A high-level language program indirectly invokes an **access method** via a get, put, read, or write statement. The execution of these statements causes an access method routine to be invoked that performs the requested input/output (I/O) operation on the indicated file.

Access methods are file-system procedures that interpret and satisfy user requests for storage and retrieval of data. In short, they are the "go-between" for a user program and a file. They can handle buffering (holding data in the main memory), blocking (placing many records into one block), and deblocking and serve as interfaces with devices [Ref. 41].

There are generally many access methods which are provided by large operating systems. These are sometimes grouped into two categories, namely *queued access methods* and *basic access methods*.

The queued methods provide more powerful capabilities than the basic methods. Queued access methods are used when the sequence in which records are to be processed can be anticipated, such as in sequential accessing (accessing ordered data). The queued methods perform anticipatory buffering and scheduling of I/O operations. They try to have the next record available for processing as soon as the previous record has been processed. More than one record at a time is maintained in the primary storage. This allows processing and I/O to be overlapped. The queued access methods also perform automatic blocking and deblocking.

On the other hand, the basic access methods are normally used when the sequence in which records are to be processed cannot be anticipated, particularly with direct accessing. Also there are many situations in which user applications want to control record accesses without incurring the overhead of the queued methods. In the basic methods, unlike the queued methods, the user must perform blocking and deblocking [Ref. 42].

During the last decade there has been considerable interest in file structures suitable for storing large dynamic files of records. Dynamic means that records can be inserted into and deleted from the file, causing the size of the file to vary. In a static file, records are not inserted or deleted and attribute values are only updated.

The file structures intended for retrieval on primary key can be divided into two classes: those based on **indexing**, which makes use of key fields to provide access to a file, and those based on **hashing**, which provides rapid access to a file.

The indexing techniques techniques have mostly developed during the sixties and at the beginning of the seventies, while hashing schemes are more conventional. We discuss hashing first, then indexing next in the following sections.

A. THE USE OF HASHING FOR DATA ACCESS

In most on-line systems, the dominant mode of file access is random, as is the case of reservation systems for airlines, hotels, and car rentals, and information retrieval systems, for libraries and stock market quotations. In these systems, both updating and retrieval are accomplished in the random mode, and there is rarely a need for sequential access to the data records. In such applications, a hashed file organization is often preferred. A hashed file organization provides rapid access to individual records, since it is not necessary to search indexes. However, here the low loading factor is traded for rapid access. In other words, in all hashed files, there are many, scattered and unused file spaces which any left unloaded with file data. Consequently, hashed files take more secondary storages than indexed files.

The basic idea behind a hashed access file organization is that the records of a file are divided among buckets, each of which consists of one or more records for storage. The major components (see Figure 36) associated with hashed files include the identifier, transformation, primary and overflow storage areas. The primary storage area is divided into a number of addressable locations, called **buckets**, which are simply physical storage blocks. Each bucket consists of one or more **slots**, where records may be stored. Records are assigned to buckets by means of a **hashing routine**, which is an algorithm that converts each primary key value into a relative disk address or bucket number. Ideally, the hashing routine that is chosen should distribute the records as uniformly as possible over the address space to be used. This provides the following two important benefits: first, collisions, which occur when two or more records are assigned to the same bucket, are minimized, and secondly, file space is utilized as efficiently as possible. The record is stored in that bucket if there is an empty slot. If all slots in the bucket are full, then the record is stored in an overflow area of the bucket.

One hashing algorithm that has been proposed that consistently performs best under most conditions is the division/remainder method, which works as follows:

(1). First, the number of buckets to be allocated to the file must be determined, and a prime number that is approximately equal to this number is selected;

(2). Secondly, each primary key value is divided by the prime number, and the remainder is used as the relative bucket address.

To retrieve a record in a hashed file, the hashing algorithm is applied to the primary key value to calculate the relative bucket address. If the record is located at its home address, then only one disk access is required. If it is in an overflow area, then two or more accesses are required. The number of accesses per record or average search length is computed as follows: average search length = (number of records * number of disk accesses) / number of records [Ref. 43].

Most files systems today support hashed files. In fact, the ease of using hashed files, of which there are six types to be discussed, is one of the major advantages of a file system. These six types are linear hashing, dynamic

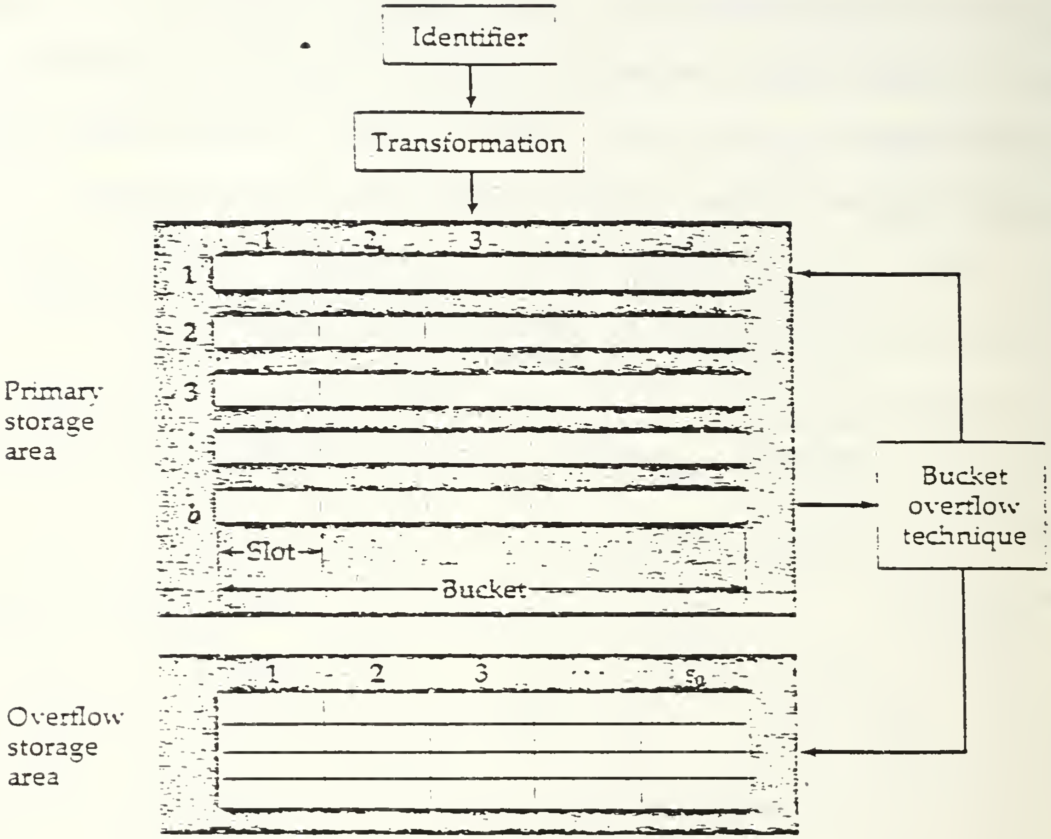

Identifier
Transformation
1
2
3
...
s
1
2
3
b
Slot
Bucket
Primary storage area
Bucket overflow technique
1
2
3
...
s_0
Overflow storage area

Figure 36. Hash Organization

hashing, *linear hashing with partial expansions*, *interpolation hashing*, *extendible hashing*, and *coalesced hashing*.

1. The Linear Hashing

The starting point for linear hashing is a traditional hash file where overflow records are handled by bucket chaining, which is the method where overflow records are stored by linking one or more overflow buckets from a separate storage area to an overflowing bucket. Each overflowing bucket has its own separate chain of overflow buckets.

An inherent characteristic of hashing techniques is that higher storage utilization results in increased search lengths, both for successful and unsuccessful searches. If the search performance of a growing file is to remain within acceptable limits, additional storage must somehow be allocated to the file. The linear hashing increases the storage space gradually by splitting the primary buckets in an orderly fashion: first bucket 0, then bucket 1, etc. A pointer p keeps track of which bucket is the next to be split.

In general terms, to implement the linear virtual hashing, starting from a file of N buckets, we need a range condition (a sequence of hashing functions) H(0), H(1), H(2),... where

H(i) K is an element of {0,1,2...N2 to the i}, i = 0,1,2...

and (for the split condition), where for each key K, either

(1) H(i+1) K = H(i) K or

(2) H(i+1) K = H(i) K + N2 to the i , i = 0,1,2,...

The hashing functions are assumed to hash randomly and hence event (1) and event (2) are equally likely. An example of such a sequence is the division/remainder technique use above where H(i) K = K mod N2 to the i.

In order to compute the address of a record at any time we must keep track of the state of the file. This can be done by two variables, let's say p and q. The variable p denotes the level of the file and counts the number of times the size of the file has doubled, while the variable q is a pointer to the next bucket to be split. A simple algorithm (see Figure 37) exists to compute both the address H of a record with key K and the splitting of the bucket.

Retrieval of a record is simple. First, H(0) K is computed and if H(0) K > 0 the bucket has not yet been split and H(0) K is the desired address. If H(0) K = 0 (i.e., a bucket has been split), then H(1) K is computed and gives the address. Finally, a method is required to establish rules for deciding when splitting of the next

```
Range Condition } $H_i : K - \{ 0,1,2 \ldots ,2^i * N-1 \}$

Split Condition } for each $k \in K$ : $H_{i+1}(k) = H_i(k)$
                                        or
                                   $H_{i+1}(k) = H_i(k) + 2^i * N$

where N is the initial number of buckets.

Process } Next  = next chain to be split
          Level = # of times the original buckets split

Access } 1. $m = H_{Level}(k)$
         2. if $m <$ Next then $m := H_{Level+1}(k)$

Split }  { a bucket is split when the storage utilization factor
           (suf) becomes larger than the predetermined threshold }
         1. create new bucket with address Next + $2^{Level} * N$
         2. rehash $k \in$ Next using $H_{Level+1}$
              k  either  remains in Next or moves to
                               Next + $2^{Level} * N$
         3. Next := Next + 1
              if Next   or = to $2^{Level} * N$ then
                        Next  := 0
                        Level := Level + 1
```

Figure 37. Linear Hashing Algorithm

bucket is to take place, called the control function. Of the several alternatives available, the following two strategies are noted, called uncontrolled and controlled splitting.

Uncontrolled means that a bucket is split whenever an inserted record is placed in the overflow area. This rule leads to low storage utilization, but good retrieval performance. If we want to achieve higher storage utilization, the control function must be more restrictive.

Controlled splitting allows splitting of a bucket to take place only when an inserted record is places in the overflow area and the overall storage utilization is above a certain pre-determined threshold. This rule obviously leads to better storage utilization but slower retrieval [Ref. 44].

Figure 38 illustrates how linear hashing works.

2. The Dynamic Hashing

The dynamic hashing scheme is based on normal hashing except that the allocated storage space can easily be increased and decreased without reorganizing the file, according to the number of records actually stored in the file. The expected storage utilization is approximately 69% at each time, and there is no overflow records. The price which must be paid for this is the maintenance of a relatively small index. If this index is available in the main storage, only one access to

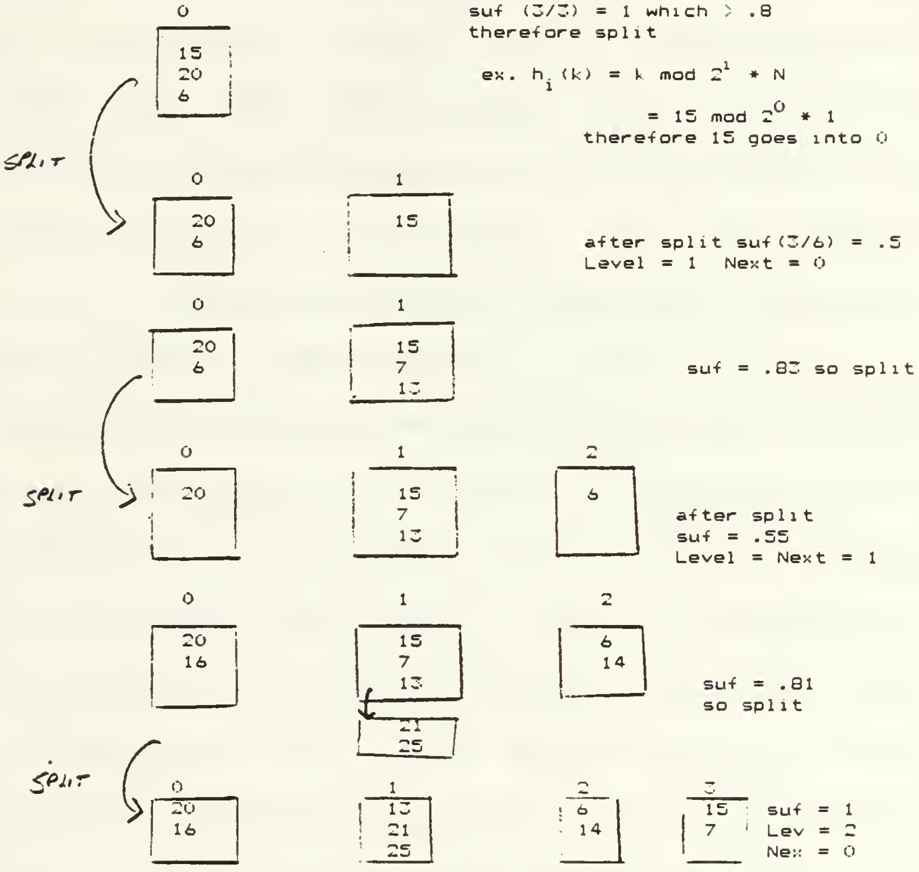
Given :

$H_i(k) = k \bmod 2^1 * N$
$N^1 = 1$
suf = .80

Insert: 15, 20, 6, 7, 13, 14, 21, 25, 16
primary page size = 3 records
overflow page size = 2 records

Solution:

Initially Next = Level = 0



suf (3/3) = 1 which > .8
therefore split

ex. $h_i(k) = k \bmod 2^1 * N$

$= 15 \bmod 2^0 * 1$
therefore 15 goes into 0

after split suf(3/6) = .5
Level = 1 Next = 0

suf = .83 so split

after split
suf = .55
Level = Next = 1

suf = .81
so split

suf = 1
Lev = 2
Nex = 0

0
15
20
6
SPLIT
0
20
6
1
15
0
20
6
1
15
7
13
SPLIT
0
20
1
15
7
13
2
6
0
20
16
1
15
7
13
21
25
2
6
14
SPLIT
0
20
16
1
13
21
25
2
6
14
3
15
7

Figure 38. Linear Hashing Example

secondary storage is necessary when searching for a record. The file organization for this scheme employs a data structure consisting of a data file in which the data records are stored, and an index to the data file. The index is organized as a forest of binary trees. The hash trees used here are closely related to binary tries. The data file consists of a variable number of buckets of fixed size. The set of records to be stored at a certain time is denoted by ri, where i = 1,2,...n. The number of records n is not fixed but may vary with time. A record Ri is assumed to contain a unique key Ki. The set of keys is denoted by Ki, where i = 1,2,...n. Each bucket in the data file has a capacity of N records.

The file is initialized in much the same way as a normal hash file. The secondary storage space is allocated for M buckets. In the index M entries are initialized, one entry for each bucket, each entry containing a pointer to a bucket in the data file. Each index entry is either an internal node, which contains pointers to its father and sons, or an external node, which contains, besides the pointer to its father, a pointer to a bucket in the data file and the number of records actually stored in this bucket. The initial buckets are said to be on the level zero. A hashing function Ho for distributing the records among the buckets is also needed. The value Ho(Ki) in this case is used to define an entry point in the index and does not refer directly to a bucket. The bucket is found by means of the pointer in the corresponding entry.

When the file is properly initialized we can start loading the file. This is also done in approximately the same way as for a normal hash file, but using the index to locate the buckets. Sooner or later a bucket will overflow, i.e. when trying to insert a record in a bucket that is already full. When this happens the bucket is split into two. The storage space for a new bucket is allocated and the records are distributed equally among the two buckets. At the same time the index is updated to depict the new situation. Additional records that would be stored in the split bucket are distributed between the two buckets. If later, one or the other of the two buckets become full, this in turn is split into two buckets. Figure 39 illustrates the structure of a dynamic file after three splits. The levels represent the hashing table or directory, the circles represent the branch nodes, the squares represent the leaf nodes, and the arrows represent the the record addresses.

When the number of stored records decreases, the allocated space can also be decreased. When the number of records in two brother buckets becomes less than or equal to the capacity of one bucket, the two brother buckets are merged into one, and one bucket can be freed. Two buckets

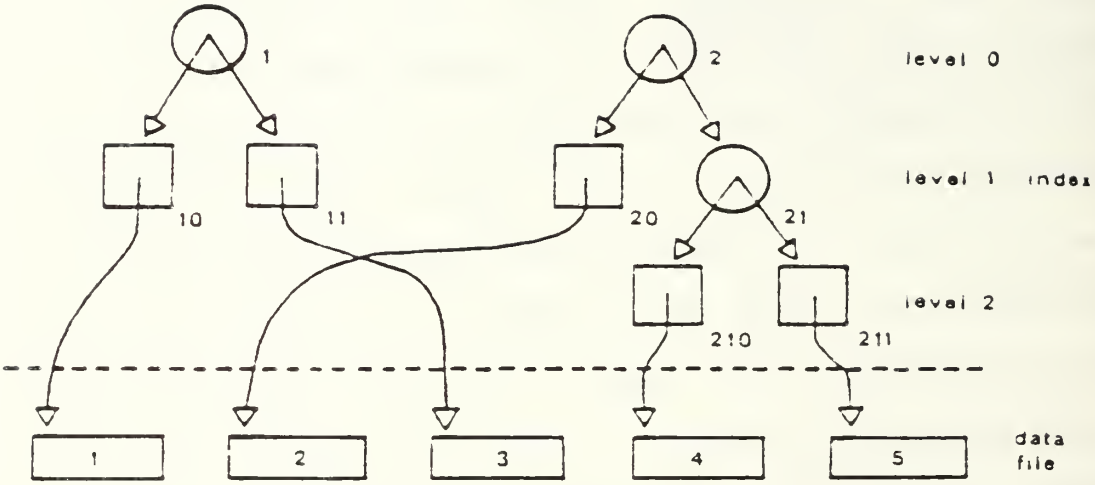

1
2
level 0
10
11
20
21
level 1 index
level 2
210
211
1
2
3
4
5
data file

Figure 39. Dynamic Hashing

are brothers if the corresponding external nodes have the same father node. At the same time the corresponding search tree is updated.

Some of the most important characteristics of this hashing scheme include that the allocated physical storage space is easily increased or decreased as required by the actual number of records stored. There is no overflow problem, since overflow records do not occur. Retrieval is fast. The retrieval of a record requires only one access to the secondary storage, provided that the forest of hashing trees is in main storage. Also, it is a simple task to find, insert or delete a record with key K, or to establish the fact that a certain record is not in the file.

Simple algorithms exist to accomplish this. Although the tree structure for this scheme is simple, it does lead to additional storage requirements. Several variants of dynamic hashing have been proposed to compensate for this drawback. The first variant is the dynamic hashing with the deferred splitting, in which the splitting is deferred until both the bucket itself and its brother are full. In other words, if the "home" bucket is full, the record is attempted to be stored in its brother bucket. If this is full as as well, or has already been split, the "home" bucket is split. This modification leads to better storage utilization but requires more complicated (and therefore slower) algorithms for searching, insertion and deletion. Also, searching is slower, since it may be necessary to search two buckets. Furthermore it is evident from experiments [Ref. 45] that deferred splitting leads to more unstable storage utilization. Thus,there is the trade-off between fast retrieval and high storage utilization.

Another variant to dynamic hashing is the linear splitting, in which the file supports a larger number of records before the index overflows on secondary storage, provided that the index is available in the main storage, experiments show that linear splitting performs just as well as the deferred splitting. The advantage over deferred splitting is that the index node size is smaller. Also, as far as the number of accesses to the secondary storage is concerned, this scheme is rather unfair, since some records are accessed very fast, while other records may require a large number of accesses [Ref. 45]. The linear hashing evolved from this second scheme.

3. The Linear Hashing with Partial Expansions

The linear hashing with partial expansions is a dynamic hashing scheme that is a generalization of the linear hashing and contains linear hashing as a special case. The main advantages of the linear hashing are retained: the file size grows and shrinks gracefully, there is no index and the maintenance and retrieval algorithms are simple. However, a significantly better search performance can be achieved. Using this technique a dynamic filer with a constant storage utilization of 85 %, for example, can be designed, where retrieval of a record requires an average of only 1.05 accesses to secondary storage.

This above scheme is based on the observation that an important characteristic of hashing techniques is that the best performance is achieved when the records are distributed as uniformly as possible over all the buckets in the file. The record distribution of the linear hashing deviates quite radically from this ideal. The load factor of a bucket already split is only half the load factor of a bucket not yet split. If a more even load could somehow be achieved, the performance of the file would be considerably improved.

The main difference compared to the linear hashing is that the doubling of the file size is done in a series of partial expansions. If this is done in two steps, the first expansion increases the file size to 1.5 times the original size, while the second expansion increases it to twice the original size (see Figure 40). Note that when examining a group of buckets, it is not necessary to rearrange records among the old buckets. It does suffice to scan through the old buckets collecting

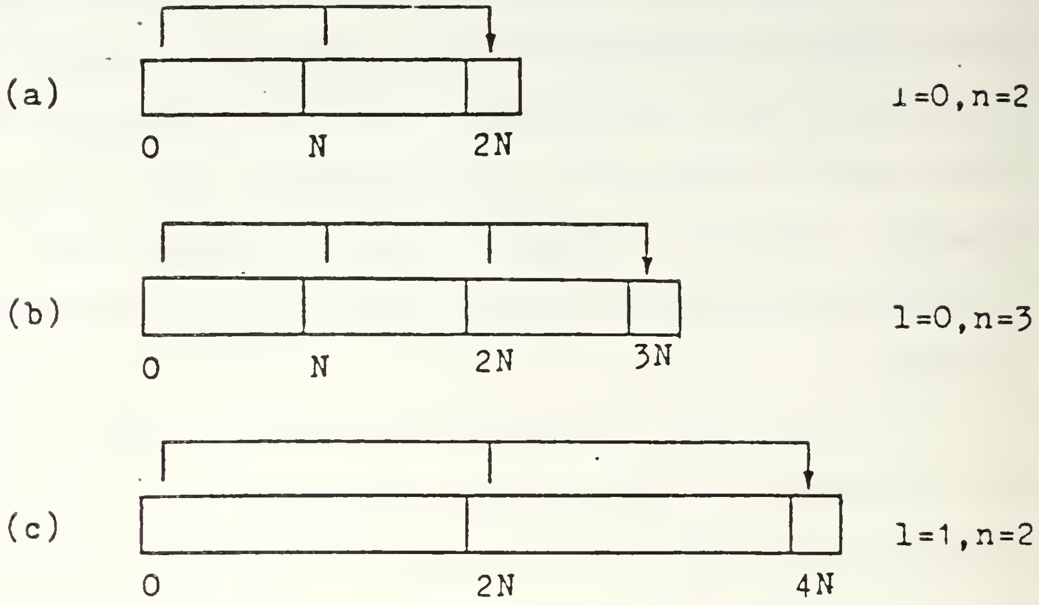

(a)
0
N
2N
l=0,n=2
(b)
0
N
2N
3N
l=0,n=3
(c)
0
2N
4N
l=1,n=2

Figure 40. Linear Hashing with Two Partial Expansions

only those records which are to be re-allocated to the new bucket. This makes it possible to make the expansion in one scan, and no jumping back and forth is necessary.

The performance will be cyclical where a cycle corresponds to a full expansion. The rate of expansion is governed by a control function, which is simply a set of rules for determining when the next expansion is to take place [Ref. 46].

The performance measures are the same as the dynamic hashing: length of successful and unsuccessful searches, cost of inserting a record , and cost of deleting a record. The cost measure is the same for all operations: the expected number of accesses to secondary storage required to carry out the operation in question. Simple algorithms exist that compute expected overflow space requirements, performance measures for the expected value at any point of a partial expansion and the average of the expected values over a full expansion. Also, contrary to what is often believed, the analysis reveals that the longest probe sequence is not expected to be very long for normal parameter combinations.

The above control function is optimal in a certain sense. Everything else being equal, there is a trade-off between storage utilization and the expected length of successful searches. The higher storage utilization is, the longer are the searches expected to be. Any one of the two factors can be controlled, but not both simultaneously. One proposal is that storage utilization is controlled by requiring that it should always be higher than, or equal to, some threshold. But once the threshold has been fixed, the expected length of successful searches will be minimized by always keeping storage utilization as low as possible. The rule above allows storage utilization to go slightly below the threshold, simply because the storage utilization after an expansion is not known before the expansion has been made. However, when the number of buckets in the file is moderately large, this rule will result in a storage utilization which, for all practical purposes, is constant and equal to the threshold [Ref. 46].

4. The Interpolation Hashing

The interpolation hashing scheme is an adaptation of the linear hashing. This scheme supports the desired operations insert, delete, update and range query. As before, the hash functions will map records to chains. Each chain will be associated with a region; however, the association of chains and regions will vary as records are inserted and/or deleted from the file. The ensemble of chains partitions the key space into disjoint regions of the equal volume.

The conditions of hash functions are to capture the order preserving nature of the scheme. The functions of the sequence H (= ho,h1,h2,...) are split functions for ho provided the following range and split conditions are satisfied for all i:

The range condition of hi is: k = { 0,1,2...2 to the i-1}

The operations insert, delete, and update each concern a single record at most. In view of the above remarks in reference to H, the operation are identical to the equivalent operation in the linear hashing. The implementation of the range query operation is unique to this scheme. The notion of range query includes both the exact match query and partial match query as special cases. That is to say, an exact match query is a range query for which u = v, where u and v are a pair of points representing the range query. A partial match query is a range query where for some components u(j) = 0 and v(j) = 1 while for the remaining components u(j) = v(j). Now, as expected, the set of records corresponding to the pair (u,v) can change with insert, delete, and update operations. The set of chains associated with the pair (u,v) however will not change [Ref. 47]. The interpolation hashing handles the range query efficiently by ordering the key attribute values. However, range query cannot be answered efficiently with a file structure that has keys scattered all over the buckets. Otherwise, this scheme does preserve the order of keys. Access and split algorithms are exactly the same as the linear hashing, only the hash function has been changed [Ref. 42].

5. The Extendible Hashing

The extendible hashing is a fast access method for dynamic files. With this technique, the user is guaranteed no more than two page or bucket faults to locate the data associated with a given unique identifier, or key. Unlike other hashing schemes, the extendible hashing has a dynamic structure that grows and shrinks as the database grows and shrinks. This approach simultaneously solves the problem of making hash tables that are extendible and of making **radix search trees** that are balanced. Radix search trees, also known as digital search trees, or tries, which examine a key one digit or letter at a time, have long been known to provide potentially faster access than tree search schemes that are based on comparisons of entire keys, for the simple reason that one comparison leads to a larger fan-out (equal to the number of characteristics in the alphabet underlying the key space). In practice, however, radix search trees tend to be used only for small files, since they often waste memory. Usually, this wasted memory occurs at the nodes near the bottom of the tree, since a trie normally contains space for many keys not in the table, because the scheme of allocating a field for each character of the alphabet at each node is better suited to representing the entire key space rather than the contents of a particular file.

The most important performance characteristic of the extendible hashing is its speed. Even for files that are very large by current standards, there are never more than two page faults necessary to locate a key and its associated information. In order to utilize this scheme, the file is structured into two levels (see Figure 41): directory and leaves. The leaves contain pairs (K,I(K)), where K is a key, and I(K) is associated information, which is either the record associates with K, or a pointer to the record. The directory has a header, in which is stored a quantity called the depth, d, of the directory. After the header, the directory contains pointers to leaf pages or buckets. The pointers are laid out as follows. First, there is a pointer to a leaf that stores all keys K for which the pseudokey K1 = H(K) starts with d consecutive zeros. This is followed by a pointer for all keys whose pseudokeys begin with the d bits 0...01, and then a pointer for all keys whose pseudokeys begin 0 ...010, and so on, lexicographically. Thus, altogether there are 2 to the d pointers (not necessarily distinct), and the final pointer is for all keys whose pseudokey begins with d consecutive ones. The depth of the directory is the maximum of the local depths of all of the leaf blocks.

In the situation where there is a single leaf block, with the local depth 1, which finally overfills, or reaches a predetermined unacceptably full level, such as 90 % full, it would split into two leaf pages, each with local depth 2. On the other hand, if a leaf block overfills, and the local depth of the leaf block already equals the depth of the directory, then as the directory doubles in size, its depth increases by 1, and the leaf page splits. This process of doubling the directory is not expensive because no leaf blocks need to be touched (except, of course, for the leaf block that caused the split and its new sibling). For example, if there are a few million keys when the directory doubles, and if the secondary storage device has a data transfer rate of around a million bytes per second, then it's straightforward to estimate that the time involved in doubling the directory (which is mainly data transfer time) would be less than a second if there were 400 keys per leaf block. Even in the extreme case of a billion keys, the time involved in doubling the directory would be less than a minute. A number of advantages accrue from the simple, intuitive

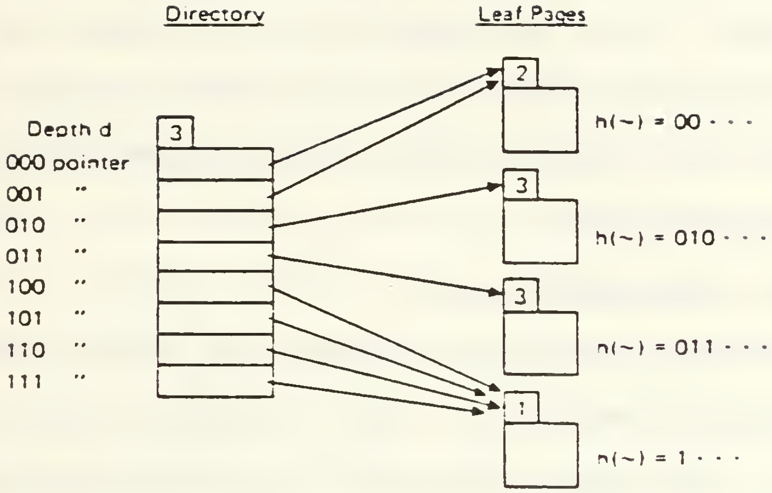

Directory
Leaf Pages
Depth d
3
000 pointer
001 "
010 "
011 "
100 "
101 "
110 "
111 "
2
h(~) = 00 · · ·
3
h(~) = 010 · · ·
3
h(~) = 011 · · ·
1
h(~) = 1 · · ·

Figure 41. Extendible Hashing

structure of the extendible hashing. The most obvious is the simplicity of the coding, thus leading to lower likelihood of bugs. Also, the extendible hashing algorithm is easily modified to accommodate individual needs [Ref. 48]. Moreover, its operating costs, as analyzed by Mendelson [Ref. 49] are fairly low. Furthermore, there is an easy, essentially one-pass algorithm for doubling the directory, that proceeds by working from the bottom of the old directory to the top of the old directory.

This technique provides an attractive alternative to other access methods.

6. The Coalesced Hashing

The term coalescing refers to the phenomenon in which a record R1 collides with another record R2 that previously collided elsewhere, and R1 is linked into R2's chain even though the two records have different hash addresses.

The coalesced hashing is a very efficient technique for storing and retrieving information dynamically. The algorithm combining storage and retrieval works as follows: Given a record with the key K, the algorithm searches for it in the hash table, starting at its hash address hash (K) and following the the links in the chain. If the record is found, the search is successful; otherwise, the end of the chain is reached and the search is unsuccessful in which case the record is inserted as follows: If the position of hash (K) is empty, then the record is stored at that location; otherwise, the record is stored in the largest-numbered empty slot in the table and linked into the chain that contains slot hash (K) (i.e., at some point in the chain after slot hash (K)).

There are several different ways to link that record into the chain. The conventional method links the record to the end of the chain that contains the slot hash (K). Another method is to insert the record into the chain immediately after slot hash (K) by rerouting pointers. This method is called early-insertion coalesced hashing or EICH, because the record is inserted early into the chain; whereas the conventional method is referred to as "late-insertion coalesced hashing" or LICH. For the standard coalesced hashing, i.e., when there is no cellar (a cellar refers to the extra space reserved for storing colliders), these methods are abbreviated EISCH and LISCH. Insertion can be done faster with the early- insertion method when it is known a priori that the record is not already present in the table, since it is not necessary to search the end of the chain.

Search times can be improved by devoting the bottom portion if the table has a cellar, in which only colliding records can be stored. Colliders that are stored in the cellar are thus protected from being collided into by records inserted later. Coalescing cannot occur until the cellar becomes full and colliders begin to be stored in the address region.

As far as performance, without a cellar EISCH has faster search performance than LISCH, but when the early-insertion coalesced hashing method is used with a cellar, the average search performance is inferior to that of LICH, since in EICH the records of a chain that are in the cellar come at the end of the chain, whereas in LICH they come immediately after the first record in the chain.

A new variant of coalesced hashing is called *varied insertion*, which combines the advantages of early insertion and late insertion. This method (VICH) is slightly different from EICH, in that the collider is inserted immediately after its hash address, as in EICH, except when the cellar is full, when there is at least one cellar slot in the chain, and when the hash address of the collider is the location of the first record in the chain. In that case, the collider is linked into the chain immediately after the last cellar slot in the chain. For the case of the standard coalesced hashing, the two methods are identical; that is, the varied-insertion standard coalesced hashing (VISCH) is the same as early-insertion standard coalesced hashing (EISCH).

The varied insertion is superior to both early insertion and late insertion with respect to search-time. VICH is search time optimum among all direct chaining methods, under the assumptions that the records are not moved once they are inserted, that for each chain the relative order of its records does not change after further insertions, that there is only one link field per table slot, and that cellar slots get priority on the available-slots list. It remains an open problem whether there are methods with faster search times than VICH if we remove the last assumption [Ref. 50].

7. A Summary

File organizations based on hashing are suitable for data whose volume may vary rapidly and for rapid data accesses at the expenses of lower loading factors. In the different variants of the hashing previously discussed, the rehashing is avoided. They do not require any thorough reorganization of the file. Further, the storage space is dynamically adjusted to the number of records being stored and there are no overflow records. Some of the techniques employ an index to the data files; others do not. The retrieval is fast; the storage utilization is low.

In order to increase storage utilization, new schemes have been discussed. In these schemes, overflow records are accepted, and the price which had to be paid for the improvement on storage utilization is a slight access cost degradation.

Dynamic hashing schemes and extendible hashing schemes employ an index to the data file; therefore, once the bucket's address has been found, the retrieval is fast. The extendible hashing is implemented by means of partitioning, in contrast, dynamic hashing schemes are implemented by means of a tree structure, which grows and shrinks more smoothly, but the index node size is larger than that of extendible hashing's index entry.

The linear hashing schemes are similar to the extendible hashing but do not employ any index. The retrieval of a record then require only one access to secondary storage. The price to be paid for this is a very low storage utilization, compared to the other schemes.

Coalesced hashing schemes have been shown to be very fast for the dynamic information storage and retrieval. Its parameters relate the sizes of the address region and the cellar. Techniques discussed are designed to tune the parameter in order to achieve optimum search times, but do have some open problems [Ref. 30].

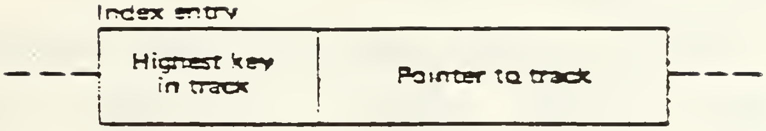
B. THE EMPLOYMENT OF INDICES FOR PRECISE ACCESS

An index is a file in which each entry (record) consists of a data value together with one or more pointers. The data value is a value for some field of some record or records in the indexed file, and the pointers identify records in the indexed file having the value for the field in the record(s). Thus, there are two types of files: The index file and the indexed file, i.e., the record file. There are also two types of records. The record in the index or index file is an index entry and the record in the record or indexed file is the data aggregate of fields. The fields whose values are kept in the index are referred to as key fields of the records. The indexed access pertains to using an index of key fields that provide the disk addresses of records stored in a file. Generally speaking, the contents of the index file are an abstraction of the file of basic source documents, i.e., it can be considered as a sort of shorthand substitution for the original document, containing only as such information as essential attributes and statistics required to satisfy the user's need. In addition, the advantage of indexing is that, although it does not access as fast as hashing schemes, it does have a very high loading factor.

It is possible to construct two types of indexes: *nondense* and *multilevel* indexes. The idea behind nondense index (see Figure 42) is that the file being indexed is divided into groups, with several stored record occurrences in each group, such that for any two groups, all the stored record occurrences in one precede all those in the other, with respect to the sequencing being imposed on the file.

The term nondense refers to the fact that the index does not contain an entry for every stored record occurrence in the indexed file. Thus, the stored record occurrences must contain the indexed field [Ref. 51].

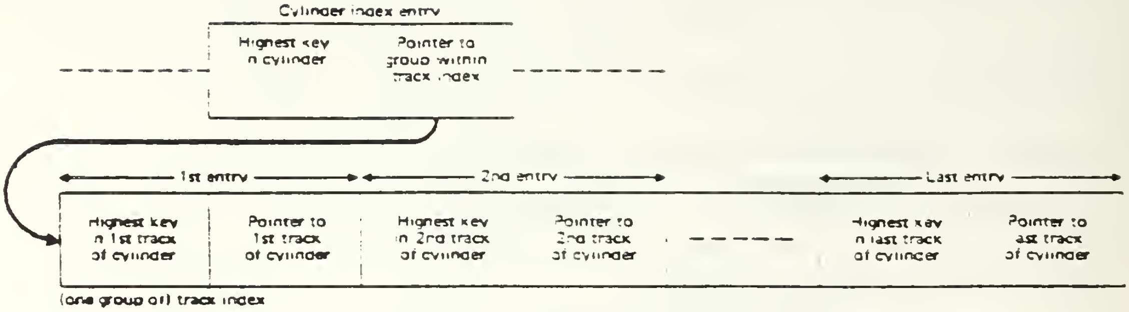
Multilevel indexing (see Figure 43), on the other hand, refers to the construction of an index to the index. Here, the indexed file is divided into groups of one track each. The track index contains an entry for each such track. The track index in turn is divided into groups, each of which consists of the entries for all tracks of one cylinder in the indexed file, and a cylinder index contains an entry for each such group in the track index. Each group within the track index is normally recorded at the beginning of the appropriate cylinder of the indexed file, to cut down on seek activity. In general, a multilevel index can contain any number of levels, each of which acts as a nondense index to the level below. An index, now, can be used in two ways. First, it can be used for the sequential access to the indexed file, in accordance to the values of the indexed field. In other words, it can impose an ordering on that indexed file. Second, it can be used for direct access to individual records in the indexed file on the basis of the value for the same key field. Other file organizations and other index techniques that are are presented are the heap, direct, primary and seconday keys, B and B+ trees, clustering, and directory hierarchy.


Index entry
Highest key in track
Pointer to track

ACCESS PROCESS:

1. Scan index for key > key required.
2. Go to indicated track.
3. Perform physical sequential track scan.

Figure 42. An Example of Nondense Indexing


Cylinder index entry
Highest key in cylinder
Pointer to group within track index
1st entry
2nd entry
Last entry
Highest key in 1st track of cylinder
Pointer to 1st track of cylinder
Highest key in 2nd track of cylinder
Pointer to 2nd track of cylinder
Highest key in last track of cylinder
Pointer to last track of cylinder
(one group of) track index

ACCESS PROCESS.

1. Scan cylinder index for key ≥ key required.
2. Go to indicated group within track index.
3. Scan track index group for key ≥ key required.
4. Go to indicated track.
5. Perform physical sequential track scan.

Figure 43. An Example of Multi-Level Indexing

1. The Heap File Organization

The most obvious and basic approach to storing a file of records is simply to list them in as many blocks as they require, although one does not generally allow records to overlap block boundaries. This organization is sometimes called a **heap**, when it is necessary to dignify it with a name. The blocks used for a heap may be linked by pointers; or, a table of their addresses may be stored elsewhere, perhaps on one or more additional blocks. To insert a record, the record is placed in the last block if there is space, or in a new block if there is no more space. Deletions can be performed by setting a deletion bit in the record to be deleted. Reusing the space of to-be deleted records by storing newly inserted records in their space is dangerous if pointers to these records still exist.

Given a key value, the record lookup requires a scan of the entire heap-organized file, or at least half the file on the average, until the desired record is found. It is this operation whose cost is prohibitive if the file in question is spread over more than a few blocks [Ref. 52].

Note that the data file has no particular order. All records are stored randomly in the file. Inserting a new record is simple; deleting a record requires a lot of data movement.

In short, the heap structure is beneficial if the records are small, and the file is temporary.

2. **The Indexed Sequential**

The development of direct-access storage devices made it feasible to transform a sequential file into a file that could be accessed both sequentially and randomly via a primary key. The **indexed sequential file organization** is such a file organization. It is probably the most popular and simplest file organization for single-key files. It is referred to as ISAM, indexed sequential access method by IBM.

Prior to discussing the above file, let's first review the sequential access. The sequential access pertains to storing and retrieving records in a one-after-the-other order. Records are generally stored in ascending or descending order by a record key. A record key is a unique unchanging piece of information such as an account number, name, or social security number. The sequential access is the only access technique used with magnetic tape drive, which are, by its design, sequential access devices.

The storage and retrieval of records in a sequential order is similar to the approach used in manual systems. Accordingly, the sequential access has traditionally appealed to organizations converting from manual to computer-based systems. This appeal, combined with the early dominance of magnetic tape as a cost-effective storage medium has led to the use of sequential-access files in most initial computing efforts.

The sequential access is used primarily in batch-processing environments. It is particularly effective when the transaction activities are evenly distributed among the records in the file. When a file is to be updated, the transactions are sorted into the same sequence of records required by the transactions. The sorted transactions are then sequentially matched against the file [Ref. 53].

In the sequential data file, records are stored in the order of primary key attribute, not necessarily physically contiguous, i.e., it could be a linked list. The insertion and deletion are straightforward if the data file is organized as a linked list. It is similar to the heap structure in operation, but its primary key can be processed more efficiently than heap organization [Ref. 52]. It is a sequential organization with two additional features. One feature is an index to provide random access to keyed records, and the other feature is an overflow area that provides a means for handling record additions to a file without copying the file. IBM refers it to the ISAM files. An ISAM file (see Figure 44) consists of three component areas: an index area, a prime area containing data records and related track indexes, and an overflow area. An access to an ISAM file can be made in either sequential mode or direct mode. When the access mode is sequential, records are retrieved in basically the same manner as they are for a keyed sequential file. The sequential accessing can begin at any record in the file. To start sequentially accessing an ISAM file at a specific record in the file, a user must specify the key value of the record. When the access mode is direct, the primary key value of the desired record is supplied by a user, and an index translates the key value into a block address. The block is accessed and brought into the main memory where it is scanned for the record containing the specified primary key value. An index is created automatically by a file as records are written into the prime area. Records are written into a prime area in the lexical order determined by the value of the primary key in each record. An index is created on the same primary key that is used to order the records in the prime area.

A number of index levels can exist in the index area of an ISAM file. The track index is the lowest level of the index, is always present, and is written on the first track of the cylinder that it indexes. A track index contains two entries for each prime track of a cylinder: a normal entry and an overflow entry. A normal entry contains two elements:

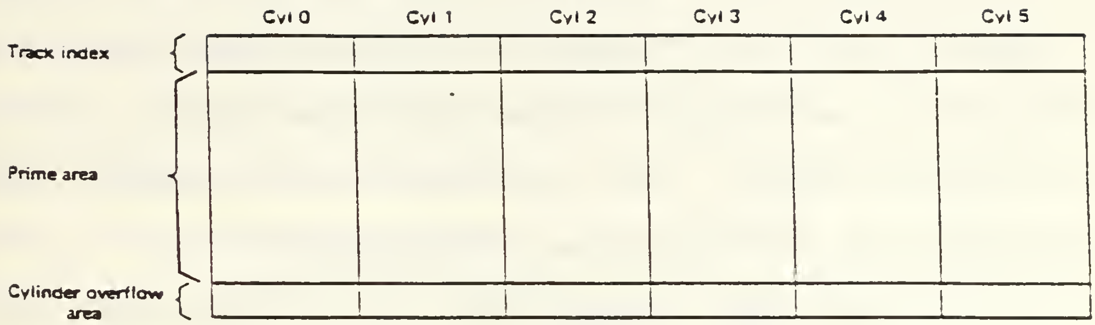

Cyl 0
Cyl 1
Cyl 2
Cyl 3
Cyl 4
Cyl 5
Track index
Prime area
Cylinder overflow area

Figure 44. Index Sequential File Organization

(1) the address of a prime track, and

(2) the key value of the last record on the track.

When an ISAM file is created, the highest key value that can appear on a prime area track is fixed, and it is maintained in the key value part of the related overflow entry. The key value of an overflow entry can change only if the file is reorganized.

The track address part of an overflow entry is initially set to contain the value 255, and it is changed when the addition of a record to the home track causes the last record on the track to be placed in an overflow area. The last entry of each track index is a dummy entry indicating the end of the index.

Just as a track index describes the storage of records on the prime tracks of a cylinder, a cylinder index indicates how records are distributed over the cylinders that make up an ISAM file. There is one cylinder index entry per track index, that is, if the data records in a file are stored on 20 cylinders, there will be 20 entries in the cylinder index. Each cylinder index entry contains the key value of the last record in the related cylinder and the corresponding cylinder address.

A final level of indexing that can exist, but does not have to exist, in this hierarchical indexing structure is the master index. Each entry in a master index contains the address of a track in a cylinder index and the key value of the highest keyed cylinder index entry on that track. The master index is used when the number of entries in a cylinder index is large, thus causing a time consuming serial search through the cylinder index for the correct cylinder containing a desired record. The master index forms the root node of the indexes used in ISAM files. The indices for ISAM files partition the prime area into small groups of records, i.e., tracks of records, so that an individual record can be accessed without accessing all the records that precede it. The problems associated with adding records to sequential files are partially avoided in ISAM files by the provision of an overflow area. Two organizations of overflow areas are possible: a cylinder overflow area, and an independent overflow area. An advantage of the cylinder overflow area is that additional seeks are not required to locate overflow record. A disadvantage is that space may be wasted, if additions are not evenly distributed throughout a file. An advantage of an independent overflow area is that less space need be reserved for overflows, and a disadvantage is that accessing overflow areas large enough to contain the average number of overflows, and an independent overflow area to be used as cylinder overflow area are filled. Updating an ISAM file may affect both the prime area and the indexes. For example, the addition of a record to an ISAM file may cause one or more of the key values in the index entries to be altered. ISAM files can be updated either in sequential or direct model. The sequential mode should be used when updates can be batched. In this case one pass is made over the file. When updates must be made on an individual basis, they should be done in direct mode.

When a record is added to an ISAM file, the prime track on which it should be placed is determined by the access method - ISAM in this case. The addition is places on the prime track, if the key value of the record is less than the key value kept in the normal entry of the related track index entry. If the record must be placed on the prime track, then a record already on the track may have to be removed and placed in an overflow area. All overflow records for a prime track are linked together in the overflow area, and a pointer to this list of overflow records is maintained in the address field of the overflow entry. The list of overflow records, if any, for each track is maintained in sorted order on the primary key. Thus, all records associated with a prime track, whether they are on the prime track or in the overflow area, are in logical sorted order. If the key value of a record to be added is greater than the key value in the normal entry of the related track index entry, then the record goes directly to the overflow area. Records are never moved from an overflow area to a prime track, unless a file is reorganized.

Records, to be deleted, on the other hand, are not physically removed from an ISAM file; instead, records to be deleted are marked by a deletion code '11111111'B, with B indicating a binary constant, in the first byte of a fixed-length record or in the fifth byte of a variable length record. If a marked record is forced off its prime track during a subsequent update, it is not rewritten in the overflow area unless it has the highest key value on that cylinder. If a record that has the same key value as a previously deleted record is later added, the space occupied by the record to be deleted may be recovered. During direct processing, marked records are retrieved and the programmer must check them for the delete code.

A record in an ISAM file can be modified in either sequential or direct processing modes. It may have to be reorganized occasionally if the overflow area becomes filled or additions increase the time required to directly locate records.

Reorganization can be accomplished by sequentially copying the records of a file, leaving out all records that are marked fir deletions, into another storage area and then recreating the file by sequentially copying the records back into the original file area. The reorganization an ISAM file with records in the overflow area usually causes new indexes to be created [Ref. 54]. In short, ISAM is an index file which contains keys to provide faster direct access to the record. Its data file is the same as sequential file. Insertion may cause an overflow, and its operation is similar to sequential file, but now the exact-match and range query can be processed faster.

3. The Direct Files

The sequential access is inefficient for batch-processing applications in which only a small proportion of the records in a file are affected by a given batch of transactions, The entire file may have to be passed to update a few records. For on-line processing, the sequential access is inadequate. The time lapse of several minutes which is generally required to locate a record sequentially is unacceptable. The sequential access fails to take the advantage of two exceptional capabilities of computer technology, namely, the speed and direct access. The direct access is an alternative to the sequential access that significantly accelerates the process of storing and retrieving records by capitalizing on both the computational speed of the CPU and the access speed of the disk drive. Disk drives are capable of directly accessing any record in a file in a matter of milliseconds. However, to access a particular record, the disk location of the record is required. The location is indicated by the address assigned to it, which is saved when the record is stored or recalculated.

In **direct files** there is a definite relationship between the primary-key value of a record and its address on a direct-access storage device. Records are stores and retrieved through the use of this relationship. Direct files allow individual records to be accessed very quickly. Since direct accessing requires the address of the specific location of a record desired, direct access requires an addressing scheme that computes a unique address for each record. Generally, the record key must be transformed into a disk storage address.

The most common approach to transforming record keys into storage addresses involves an arithmetic procedure that generates "random addresses" from record keys. There are several randomizing algorithms. The most common algorithm involves the generation of addresses by dividing the record key by positive prime number, usually, the prime number is the largest prime number that is less than the number of available addresses. The remainder of a division operation is used as the address locator. A randomizing algorithm always generates the same address for a particular key. Therefore, given the key to a record, the computer can calculate the disk address and then access the record in a matter of milliseconds.

Unfortunately, as in the case of hashing, occasionally, a random-address generator generates the same address for two or more keys. The second and succeeding records with duplicate addresses are referred to as synonyms (as collisions in hashing). When a synonym occurs, the record having the duplicate address can be stored in a location next to other synonyms of the record stored at the computed address, or it can be stored in a general overflow area. In either case, if a desired record, which is determined by checking the key stored in the record, is not located at the computed address, a sequential search of synonyms is invoked until the desired record is located. This sequential search slows processing slightly. A good randomizing algorithm generates few synonym (collisions) for a particular set of keys, further analysis and modification of the randomizing algorithm is needed.

Another drawback of a direct-access technique is that, by design, it usually leaves large gaps between records on a disk. This results in wasted disk space, as in the case of hashing. Some of the gaps may be consumed by synonyms, but considerable wasted space still remains. An offsetting advantage is the incredible speed with records can be accessed, regardless of the size of the file. A very important concept to grasp with respect to direct accessing is that the physical locations of records bear no relationship to the logical view of the data. The random generation of addresses physically scatter records throughout the disk such that, without knowledge of the randomizing algorithm used to transform the keys, locating a record requires a sequential search through a non-sequentially ordered file. The use of a randomizing algorithm is an extraordinary deviation from the way that files are maintained manually. However, it is a highly suitable technique for computer-processed files. Any one record of several thousand or even several million can be located instantaneously at the expenses of some disk-space wastage [Ref. 53].

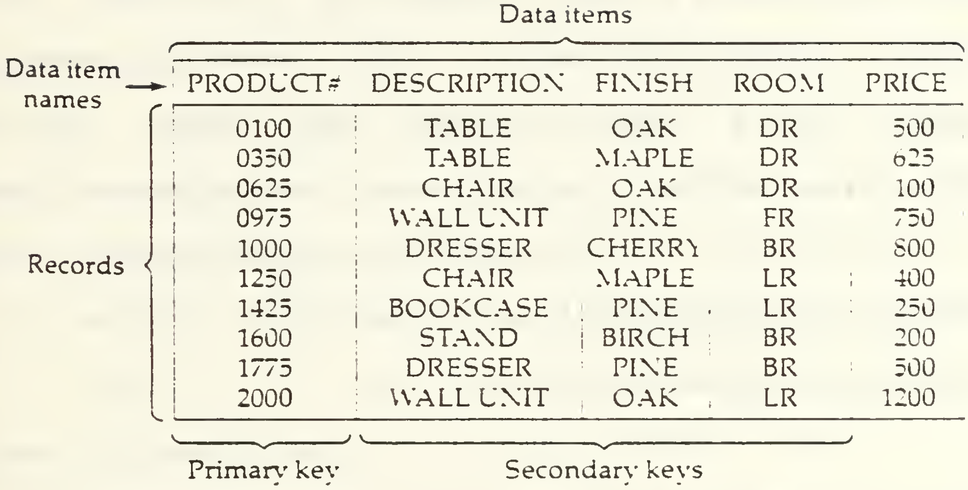
4. Primary and Secondary Keys

The run-time performance of a file system is influenced by the software to organize and subsequently access the requested data. Fast-access methods can generally be designed when all logical conditions are expressed in terms of primary keys alone, i.e., all access requests are to single records via their primary keys. It is much more difficult to design fast-access methods when logical conditions require secondary keys, that is, when access requests are to set of records. A primary key is a data item that uniquely identifies a record. The primary key of a record corresponds to the identifier of a real-world entity. As with identifiers, there may be several possible or candidates primary keys for the same record. Also, two or more data items may be required to identify a record. A secondary key is a data item that does not uniquely identify a record but identifies a number of records in a set that share the same property. For example, the data item MAJOR might be used as a secondary key for STUDENT records. Of course, this data item does not identify a unique record; for example, many students will have business as a major. However, the secondary key does identify a subset of students who are business majors. Secondary keys arise when data are referenced by categories.

Not all secondary keys need to be indexed, but before a database designer can decided which indexes to create, all secondary keys must be identified. When all data processing is known in advance, then computer program specifications provide an excellent source to identify secondary keys. Figure 45 illustrates some general guidelines for identifying secondary and even alternative primary keys [Ref. 53].

5. B-Trees and B+ Trees

If an index is helpful in storing and searching through data records, then it is possible for one index to help to organize another index. ISAM and other file organizations, as well as a host of other data structures, are all based on this approach of indexing indexes. This type of hierarchy of data and pointers to data is generalized by the tree data structure. Trees can be used to organize data directly or organize indexes into data.


Data items
Data item names
Records
Primary key
Secondary keys

PRODUCT#	DESCRIPTION	FINISH	ROOM	PRICE
0100	TABLE	OAK	DR	500
0350	TABLE	MAPLE	DR	625
0625	CHAIR	OAK	DR	100
0975	WALL UNIT	PINE	FR	750
1000	DRESSER	CHERRY	BR	800
1250	CHAIR	MAPLE	LR	400
1425	BOOKCASE	PINE	LR	250
1600	STAND	BIRCH	BR	200
1775	DRESSER	PINE	BR	500
2000	WALL UNIT	OAK	LR	1200

Figure 45. An Example of Primary and Secondary Keys

A tree data structure has the property that each element of the structure, except the root has only one path coming in,(that is to say that there is only one pointer that points to any given element), but they may be zero or many paths coming out of an element.

The tree structure described above are binary search trees and multiway (m-way) search trees. A binary tree is one of the best known and most frequently used data structures for organizing data that are stored entirely in the main memory while being processed. While binary trees have seldom been used for organizing large files in two-level storage, the multiway tree, which is essentially a generalization of the binary tree, has received widespread use for organizing indexes for large files in external memory.

The principle reason for the frequent use of the multiway search tree and the infrequent use of the binary search tree in two-level storage involves the number of accesses to the external storage to locate a record with a specific key value.

The B-tree, where B stands for balanced, (meaning all leaves are the same distance from the root), is a multiway-search tree with restricted growth (see Figure 46). B-trees guarantee a predictable efficiency that many other types of trees do not. A B-tree of order m is a tree that that satisfies the following properties:

(1) Every node has less than or equal to m sons.

(2) Every node, except the root and leaves, has at least m/2 sons.

(3) The root has at least 2 sons unless it is a leaf (terminal node).

(4) All leaves are at the same level and only contain pointers to the actual data records, i.e., carry no information.

(5) A nonleaf node (an internal node) that has k children will contain k-1 keys.

The effectiveness of a B-tree search is determined by the shape of the tree, and the shape of the tree is determined by the order, m. When m is small, a tree is tall and narrow, and when m is large, a tree is short and bushy. The maximum number of nodes, K, that must be accessed during a B-tree search is

$$K \le 1 + \log_{(m/2)} ((n+1)/2)$$

where m is the order and n is the number of key values in the tree.

Since it is preferable to use a large value for m, if m = (n+1), then only one level exists in a tree; this choice of m, however, is not reasonable if a tree is too large to fit in the main memory. When a value for m is selected, the primary objective is to minimize the total amount of time required to search a B-tree for a key value KV. This time has two component:

(1) the time required to access a node in the external storage, and,

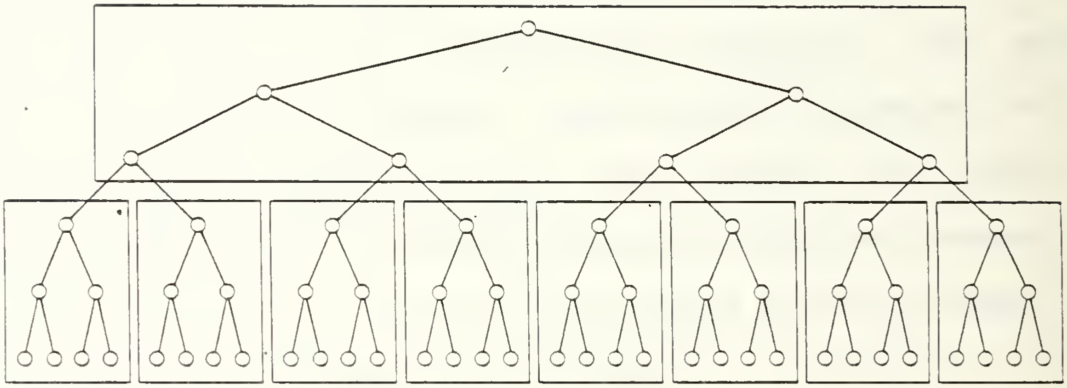


Figure 46. An Example of a Binary Tree Organization

(2) the time required to search this node,
in internal memory, for KV. It turns
out that there is a value of m, m1, for
which the search time is a minimum.
For values of m exceeding m1, the
total amount of time required to search
a B-tree increases [Ref. 54].

Searching a B-tree for a specified key value is as follows. A node, starting with the root node, is brought into the main memory and searched, possibly using a binary search for the given argument key value among the key values K1,K2,...KJ. If the search is successful, then the desired key value is located, but if the search is unsuccessful because the argument key value lies between Ki and K(i+1), then the node pointed to by Pi, which points to a subtree holding key values between Ki and K(i+1), is retrieved and the search continued. The pointer Po is used if an argument key value precedes K1, and P1 is used if an argument key value follows KJ in sorted order. If Pi = null, the search is unsuccessful.

The insertion process for B-trees is relatively simple; each terminal node corresponds to a place where a new key value may be inserted. Its algorithm is rather simple and straightforward. In general, a key value into a B-tree of order m with the terminal nodes at level L is placed in an appropriate node on level L-1. If this node now contains m key values, then it must be split into two distinct nodes. For example, if a node after the insertion of a new key value looks like

Po K1 P1 K2 ... P(m-1) Km Pm then it is split into two nodes

Po K1 P1 K2 ...K((m/2)-1) P((m/2)-1) and P(m/2) K((m/2)+1 ... Km Pm and the key value K(m/2) is inserted into the father of the original node. This insertion may cause a father node to contain m key values, and if so, it is split in the manner illustrated above. If a root node must be split (a root of course has no father), then a new root node is created containing the single key value K(m/2). The tree becomes one level taller in this case. Thus, a B-tree grows upward from the root top instead of downward from the leaves. The procedure described above for inserting new key values into a B-tree is exactly the procedure used to create a B-tree [Ref. 43].

The deletion of a key value from a B-tree is more complicated than inserting a new value into a B-tree. The deletion of a key value on level L-1 simply causes it to be erased from a node. When this erasing makes a node too empty , that is, underflow occurs, the right or left brother is examined and key values are moved from the brother until both nodes have approximately the same number of key values. The key values are not moved directly from the brother to the underflow node, instead, the preceding key value in the parent mode is moved to the underflowed node and the preceding key value in the brother replace the key value in the father node. A delete operation that results in underflow fails only if the brother is minimal full [Ref. 54].

In short, as far as its operation, B-trees are similar to the indexed-sequential file, but with better performance for insertion and deletion. Also, it does not require a separate index file. However, the scanning of subfiles is not efficient as the indexed-sequential file.

A B+tree is represented by nodes which are implemented by different blocks in the file. The free-block list (FBL) is used to maintain a list of free blocks for dynamic (space) management of blocks. Initially, all blocks of the file are placed on the list by initializing the FBL fields accordingly. Subsequently, the block allocation from the list takes place at the beginning of the list. Once a block is allocated to the running process, the same field is used for a different purpose if it happens to be a sequence block. It is used to store the block number of the sequence block next in the lexicographical order. If the block is an index type, the field is not used any longer. The second field or BTYPE, indicates the type of the block (node), whether it be an index or sequence. Each B+tree in the index file is identified by its root node, and the root node number is stored in the master index, in which there is an entry for each secondary key and the primary key defined for the file.

Research has been undertaken by Koymen [Ref. 55] to design and implement a B+tree-based keyed sequential access method (KSAM). KSAM provides primary and secondary access for either direct or sequential processing. Primary access to a data file requires three levels of indexes: super, master, and primary indexes. Secondary access requires an additional index level, i.e., is secondary indexes. The superindexes and master indexes are transparent to the user and are used solely by the system. The primary index is organized as a B+tree, containing proper linkage to the respective data files. In the implementation of secondary indexes, a file is used to store accession lists, a term applied to the records in a secondary index, which contain pointers used in accessing the data records. Each secondary index is in turn organized as a B+tree containing proper linkages to accession-list files. Thus, linkage from the B+tree of a secondary index to the respective data files is provided via the accession-list file. Finally, another file is used to represent all the B+trees associated with a data file. Thus, three files suffice for the implementation of a KSAM data file and its associated indexes. The implementation schema organizes each of the three files as a direct-access file. Thus the high popularity of direct-access files makes the implementation possible in almost any programming language [Ref. 55].

In short, then, a B+tree has the same index file as the B-tree (with key values only), has similar data file as the sequential file, and as far as its operations, it has the benefits of both the indexed-sequential file and B-tree index structure.

6. Clustered Files

The notion of the classification has been used for centuries in many disciplines of the social and natural sciences. The classification involves placing a set of objects into classes or "clusters" in such a way that the objects within a class are more similar to each other than they are to objects outside the class. The similarity between objects is defined in terms of known properties of the objects. Such a grouping is referred to as a "clustering" or a classification. A file in which documents that exhibit same sets of index terms are grouped into clusters is called a clustered file. To facilitate searching in a clustered file each cluster is identified by a representative called the "centroid". A search is usually carried out by first comparing a query with all the cluster centroids; then, for those centroids exhibiting a sufficiently high similarity with the query, all objects in the corresponding clusters are examined and those that have a sufficiently high query document similarity are retrieved. This approach is based on the assumption that documents in the same cluster are of interest to the same user and would therefore be requested jointly.

Since the objects within a cluster are retrieved jointly, it is desirable that they be kept in close proximity within the physical storage. A simple scheme by which objects of the same cluster can be kept close to each other is to store objects by clusters, so that each object is stored as many times as the number of clusters in which it appears, and all objects in the same cluster are stored consecutively. This scheme is called "cluster-inverted" organization.

Clearly, if the clusters are pairwise disjoint, that is, the classification is a partition, the above scheme has no redundancies. This is the case of formatted databases. However, if clusters are allowed to overlap, which is the case of textual databases, texts that are pertinent to more than one cluster have to be stored repeatedly. One solution to this problem is to store objects (i.e., texts) of each cluster in contiguous locations while minimizing redundancy. This is accomplished by characterizing conditions under which an arrangement without redundancy exists. Since the records are to be jointly retrieved in the response set of any query, the property is termed the consecutive retrieval property, CRP. There are two types of clustered files; a single-level clustering file to have CRP has been developed, in the form of a linear time algorithm to test CRP for a given clustered file and to identify the proper arrangement of objects, if CRP exists. Experimental results by Deogun, Raghavan and Tsou [Ref. 56], indicates that the algorithm generates minimum redundancy organization most of the time. Moreover, on the average, the near optimal solutions show approximately a 50 percent improvement in the amount of redundancy over the the worst case organizations.

In the case of multilevel clusterings, the problem of minimizing the number of different arrangements of objects that have to be stored has been investigated with the following results. It is shown that for any nonoverlapping multilevel clustering, there exists an arrangement of the objects such that every level clustering has CRP with respect to the arrangement. Similar results have been obtained for certain classes of overlapping multilevel clusterings [Ref. 56].

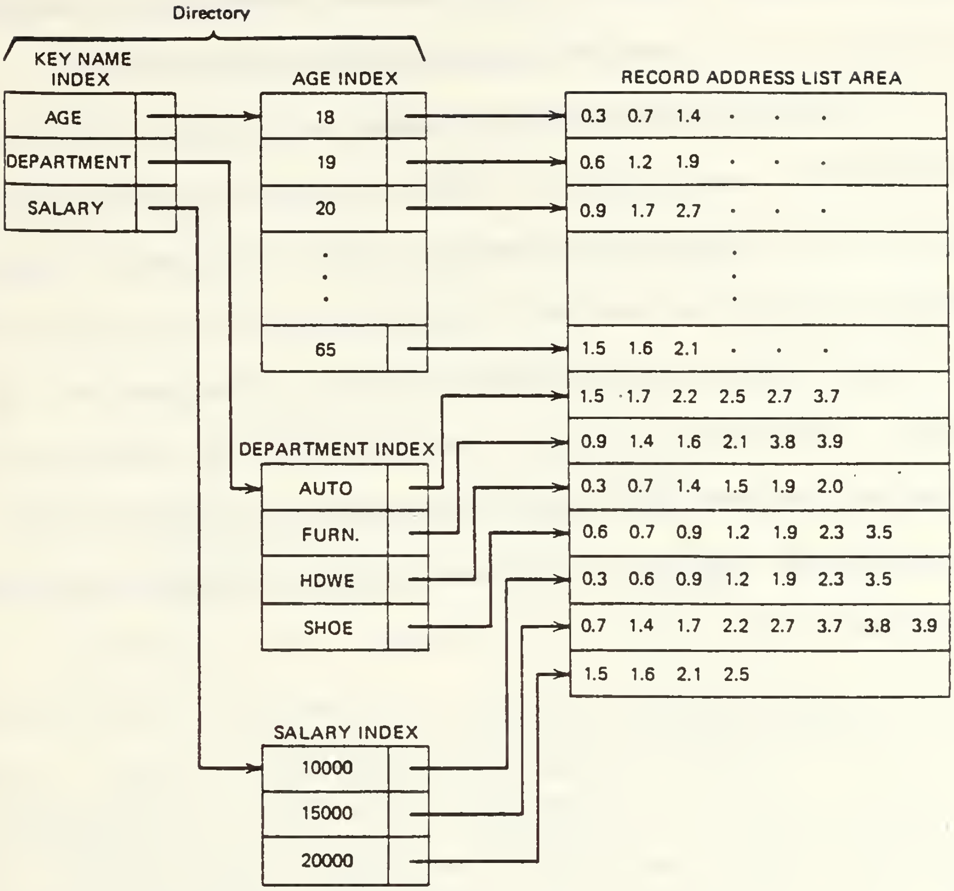
7. The Directory Hierarchy

A directory performs an important function within a file system called mapping. This particular phenomenon permits a user to create name spaces and to store (retrieve) data in (from) them. Name mapping converts a symbolic file name into a physical file address that identified where the file is stored.

Each user may have a directory of his own files and may create subdirectories to contain groups of files conveniently treated together. A directory normally behaves like an ordinary file. A file system controls access to the contents of directories; however, anyone with the appropriate authorization can access a directory just like any other file. In designing a system of file directories, it is natural to think in terms of a hierarchy with the entries in the higher levels of a directory being other directories. The entries in the lower levels are a mixture of files and directories. The directory entries in a hierarchically structured directory can contain either system directories or user directories. Data files are at the lowest level (see Figure 47).

The most important system directory is the master directory (or root directory). Files created by users are usually located by tracing a path through a chain of directories, starting with the master directory, until the desired file is found.

An interactive user or a program running on behalf of a user references a specific file via a symbolic file name. A symbolic file name is usually in the form of a path name that is a sequence of names separated by some specially designated character such as a period or slash. In the simplest case there is one to one correspondence between symbolic file names and files.


Directory
KEY NAME INDEX
AGE
DEPARTMENT
SALARY
AGE INDEX
18
19
20
65
RECORD ADDRESS LIST AREA
0.3 0.7 1.4 . . .
0.6 1.2 1.9 . . .
0.9 1.7 2.7 . . .
1.5 1.6 2.1 . . .
1.5 1.7 2.2 2.5 2.7 3.7
0.9 1.4 1.6 2.1 3.8 3.9
0.3 0.7 1.4 1.5 1.9 2.0
0.6 0.7 0.9 1.2 1.9 2.3 3.5
0.3 0.6 0.9 1.2 1.9 2.3 3.5
0.7 1.4 1.7 2.2 2.7 3.7 3.8 3.9
1.5 1.6 2.1 2.5
DEPARTMENT INDEX
AUTO
FURN.
HDWE
SHOE
SALARY INDEX
10000
15000
20000

Directory and address list for the PERSONNEL file on the keys AGE, DEPARTMENT, and SALARY.

Figure 47. An Example of a Directory

However, in more advance file systems the same file may appear in several directories under possibly different names; that is, a single file may be shared by two or more user groups, with each group having a different path through the directories to the file itself.

Directory structures imply that the output of a directory search for a file is the file itself. This is slightly misleading, because the terminal nodes of the hierarchical structure, rather than containing the file, normally contain an object commonly referred to as a file descriptor or a pointer to a file descriptor. A file descriptor contains information concerning the physical location of the file and the physical characteristics of the file [Ref. 54].

C. THE STRUCTURE OF DATA FOR RETRIEVAL

The primary objective of file organization is to provide a means for record retrieval and update. The update of a record involves its deletion, changes in some of its fields or the insertion of an entirely new record. Certain fields in the record are designated as key fields or search keys. Each record includes at least a search key which is used to generate the index of the file. A combination of search keys specified for retrieval is termed as a query. The simplest structure of data is the

indexed sequential search, which has already been discussed. For more elaborate retrievals, the structures, include **multilist**, **inverted file**, and **cellular multilist**, as well as **hybrid** and **ring structured files**. They are discussed herein.

1. The Multilist File Organization

The **multilist file organization** consists of a directory file containing index entries, and a data file. An index entry for a key value consists of the key value, a pointer to the list of records in the data file, containing the key value, and the number of records in the list.

Figure 48 illustrates a multilist file indexed on the DEPARTMENT and SALARY keys. The format of each data record in a multilist file consists of two segments [Ref. 41]. Segment one consists of one or more key/key-value/pointer triples, in which the pointer points to another record containing the same key/key-value pair. The second segment contains the values of the nonkey data items, if any. For example, a record in Figure 48 has the format:

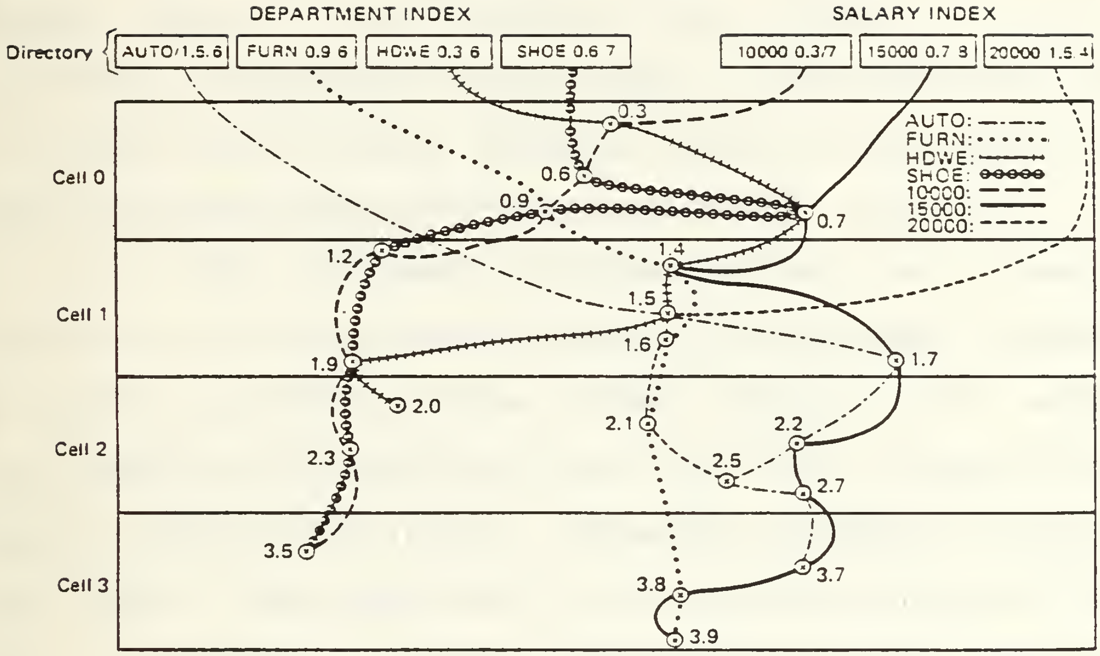
DEPARTMENT/AUTO/1.7,
DEPARTMENT/HDWE/1.9,
SALARY/20000/1.6,
Nonkey data item values.

a. Answering Queries

A technique that can be employed for answering queries within this file organization is to minimize the number of records that must be searched. This is especially important, since some lists may be lengthy and require longer search times. For example, a query to retrieve the records of all employees that work in an auto department and have a salary of 15,000, is evaluated as follows. First, the two key values in the proper indexes DEPARTMENT/AUTO and SALARY/15000 are evaluated. Secondly, the index entries for AUTO and 15000 are examined to determine their respective list lengths. Then, the shortest list is examined, which in this case is AUTO (it has 6 records as compared to 8 for 15000). Since the records with addresses 1.7, 2.2, 2.7, and 3.7 in Figure 48 have both of these occurrences, they therefore satisfy the query. For query conjunctions, we search only the shortest list.

b. The Query Cost

The cost to process a query for the multilist organization is measured in terms of the time required to decode all key values in the query and to retrieve data records. The query cost is therefore defined as Q = LT, where Q is the query cost, L is the shortest list length in a query, and T is the average time to access a record, which includes the seek time, latency time and data transfer time. Thus, when the list lengths associated with key values in the terms in a product are small, the cost for query processing is minimized.


DEPARTMENT INDEX
SALARY INDEX
Directory
AUTO/1.5.6
FURN 0.9 6
HDWE 0.3 6
SHOE 0.6 7
10000 0.3/7
15000 0.7 8
20000 1.5 4
Cell 0
Cell 1
Cell 2
Cell 3
AUTO:
FURN:
HDWE:
SHOE:
10000:
15000:
20000:
0.3
0.6
0.9
0.7
1.2
1.4
1.5
1.6
1.7
1.9
2.0
2.1
2.2
2.3
2.5
2.7
3.5
3.7
3.8
3.9

Multilist file indexed on keys DEPARTMENT and SALARY.*

Figure 48. An Example of a Multilist File

c. Updating Multilist Files

Lists can be ordered or unordered. Adding a record to an ordered list requires that the record be inserted in a specific position, while for an unordered list, at the head of the list, thus avoiding the need to traverse the list.

Updating of multilist files involves either key value addition, whole record addition, or deletion. Regardless of the type of addition, whether whole record or new key value, one or more of the index entries are also updated. When the lists are not ordered, there exist a simple algorithm in which a new record can be easily placed at the logical head of each list of which it is to be a member. This is true both types of additions. For example, to add key values to an existing record, such that an employee who works in HDWE department shares his work time between the HDWE and FURN departments. The employee's record must be updated by adding the key value FURN to his record. Adding a new key value to a record implies that the record must be added to the list of records indexed by the new key value. The simple algorithm can be used to add one or more key values to a record.

Deleting a key value from a record is essentially the same as deleting the record from the list indexed by the key value. Key value and whole record deletions can be accomplished by using a simple deletion algorithm, which simply adjusts pointers. If, however, deletion implies physically removing a record from lists, and the retrieval system performs real-time updating and retrieval, then bi-directional lists should be considered to represent the multilist. A bi-directional list allows deleting records without traversing the list to locate its predecessor record. Although bi-directional lists allows deletions of records to be done more rapidly, there is the storage overhead of an additional pointer element for each key value in a record.

2. The Inverted File Organization

Unlike the multilist files, where records are linked together with pointers kept in the individual records, the pointers in an inverted file are removed from the individual records and kept in separate list, called inverted lists. An inverted file consists of two components, a directory and a data file (see Figure 49). The variable-length inverted lists of pointers corresponding to key values are kept in the directory. Thus, when a key value is decoded in the directory, the record address list is immediately available and no additional access is required to move it into the main memory. It is important to note that the directory should be kept as small as possible so that updating can be performed quickly and easily, and that it can be kept in its entirety on a fast storage device. However, unlike the directory of the multilist file which tends to be small in size, the directory of the inverted file tends to be larger in size. The size of a the directory can be controlled by limiting the number of data items on which a file is inverted. A partially inverted file stores the record addresses associated with all values of certain (i.e., keys) but not all data items. A completely inverted file is one in which every data item is treated as a key and the record addresses associated with every key value are stored in inverted lists. In this case, the directory subsumes the file. There is no need of keeping the data file any more.

a. Answering Queries

The inverted file organization allows rapid access to records based on any key. Queries can be determined by accessing and manipulating the inverted lists of record addresses prior to accessing any data records. This advantage is possible because the pointers to records indexed by a key value are maintained in an inverted list rather than in the data records. For example, to retrieve the records of all the emplovees that work in an AUTO department and have a salary of 15000 dollars, the key values AUTO and 15000 are decoded in the proper index and produces the address of the list of data records indexed by the key value, DEPARTMENT/AUTO, and SALARY/15000. The address of the list of data records are pointers in the inverted lists for key values AUTO and

DEPARTMENT INDEX

AUTO	1.5	1.7	2.2	2.5	2.7	3.7	
FURN	0.9	1.4	1.6	2.1	3.8	3.9	
HDWE	0.3	0.7	1.4	1.5	1.9	2.0	
SHOE	0.6	0.7	0.9	1.2	1.9	2.3	3.5

SALARY INDEX

10000	0.3	0.6	0.9	1.2	1.9	2.3	3.5	
15000	0.7	1.4	1.7	2.2	2.7	3.7	3.8	3.9
20000	1.5	1.6	2.1	2.5				

Inverted lists for the data records

Figure 49. An Example of an Inverted List

15000, respectively. The two inverted lists are moved into the main memory, and the intersection of these two lists is performed. The addresses in the intersection list, namely addresses 1.7, 2.2, 2.7, and 3.7, are the addresses of the records that are to be retrieved and that will satisfy the query (see Figure 48).

b. The Query Cost

In the inverted file, the cost to process a query is the sum of the cost to decode all key values in the query, plus the cost to access all inverted lists, one per key value, plus the cost to process the inverted lists, plus the cost to retrieve the data records that satisfy the query. For the inverted organization, there are L/N records, where L is the shortest list length in a query and N is the number of data record addresses per record, accessed for each of the average number of terms in a single query, designated by T. Thus, the time required to retrieve the T inverted lists involved in the list intersection is L/N * T * A, where A is the average time to access a record and move it to internal memory. Whereas for a multilist every record in the shortest list, designated S, must be accessed, for the inverted organization only those pS records, where p is the ratio of the number of records that satisfy a query to S, must be accessed. Therefore, the query cost is defined as Q = (pS + (L/N * T)) * A. It is important to note that the larger the number of records per inverted list, the larger the amount of time to access the inverted lists.

c. Updating Inverted Files

Updating an inverted file is more involved because the inverted lists must be updated. For this reason, the inverted organization is most useful for retrieval when the update volume is relatively low compared to the query volume. By performing intersections and unions of inverted lists, the inverted file system can provide exact statistics about the records having certain key values. Whereas multilist file systems can only provide upper bounds or approximations of the record numbers of those records. Whole record and key value addition and deletions are accomplished by straightforward algorithms.

3. The Cellular Multilist File Organization

The cellular multilist organization is derived from the multilist organization. Since the performance of a multilist system suffers when lists are lengthy, cellular multilist organization is an attempt to arrange the records for more optimal retrieval with shorter lists. The length of each list is restricted to the storage-cell size so that records in the list do not extend beyond cell boundaries. A cell can be considered as a track or cylinder of a disk.

Each index entry for a cellular multilist file consists of one or more list head pointer/list length pairs.

There exists one such pair for each cell containing records indexed by a key value. The directory for this type of files is similar to multilist file, except that it is larger. Figure 48 illustrates the corresponding DEPARTMENT index, organized for a cellular multilist file, for the same data for multilist file. The index entry for key value AUTO has three list head pointer/list length pairs, each one corresponding to a list in a single cell. That is, the AUTO list in Figure 48 is subdivided into three shorter lists, one of length two in cell 1 with the head at address 1.5, one of length three in cell 2 with the head at address 2.2. and one of length one in cell 3 with the head at address 3.7.

a. Answering Queries

The cell concept is used to provide good response time. For example, to retrieve the records of all employees that work in an AUTO and HDWE department, note that the records having key value AUTO reside in cells 1, 2, and 3, and records having key value HDWE can be found in cells 0, 1, and 2. The only records that can be common to both lists are located in cells 1 and 2. The AUTO list for cell 1 is traversed, since the length of the list in cell 1 is shorter (length is two for AUTO and three for HDWE). Each record is then examined of the existence of the key value HDWE. Record 1.5 belongs to both lists. In cell 2, the HDWE list is the shortest so each record in it is accessed and examined for the occurrence of AUTO. Since there are no records in cell 2 common to both lists, only record 1.5 satisfies the query.

Updating cellular multilist files is essentially the same as updating multilist files.

4. Comparison of Multilist and Inverted Files

Figure 50 illustrates the advantages and disadvantages of multilist and inverted list files.

5. Other File Organizations for Retrieval

Other ones [Ref. 54] include the hybrid list file organizations and ring structured file organizations.

The hybrid list file organization, as its name implies, is a hybrid between a multilist and an inverted list organization. Hence, this hybrid file is organized in such a way as to minimize the system search effort in answering queries with shorter lists of records, by utilizing special cases of multilist and inverted list file organizations. Note that, for longest list of records, the multilist structure is optimum, while for shortest list of records, the inverted list structure is practical.

The hybrid list file organization is therefore defined as follows: A hybrid list organization of parameter L is a list structure that stores a set of pointers to records containing the key value as a

Organization	Advantages	Disadvantages
Multilist	Easily programmed. Conjunction queries are efficiently handled for short lists. Easily updated since complete reorganization of lists is avoided. Good for simple and range queries.	Conjunctive queries are inefficiently handled for long lists. The number of records that satisfy a query bears no relation to the number of records accessed.
Inverted	Low response time for conjunctive queries. Efficient use of storage space if key values are removed from data records. Satisfactory for real-time retrieval.	Updating is complex since the inverted lists are variable in length and must be ordered. Work space in internal memory is required to perform the logical processing of inverted lists.

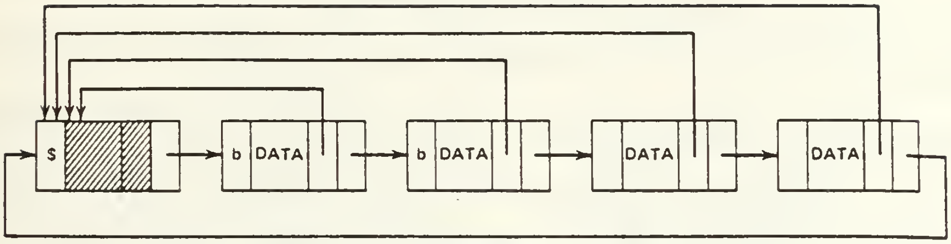
Figure 50. Comparison of Multilist and Inverted File Organization

separate inverted list if the query with a list of records is greater than L; otherwise, the set of pointers is embedded in the data records as a multilist organization. In answering queries, the retrieval is done in straight inverted or multilist fashion, depending on the size of the list of records.

The cost of updating for hybrid file organizations lies somewhere between that for the two pure organizations since it is easier to update a multilist file than an inverted file. The more lists that are stored as multilists, the easier it is to perform updates.

A ring structured file organization, on the other hand, is a linear list in which the pointer in the last record points back to the first record called the starting record of the ring. In a ring structure once an arbitrary record is accessed every other record in the ring also becomes accessible. A ring structure file consists of three elements, a value of the data item, (which is usually associated with each pointer to specify in which rings a record is an element), data, and a pointer to the next record. Figure 51 illustrates a ring structure, in which, the value of the data item is "s" for the starting record and "b" otherwise.

One of the primary advantages of a ring structure is that any record can be accessed starting at any point. An important use of ring structures is, thus, to represent classifications (types) of data. All records having the same classification (type) belong to the same ring. Associated with each record within a class (type) may be a subclass (subtype), and this subclass is represented by a ring of records. Such a classification scheme can be represented by use of multiple ring structure, in which multiple rings pass through a record with the records in each ring logically related. Figure 52 illustrates such a structure, corresponding to the same data as in Figure 48. The values of the data items, MD, MS, D, and S are used to designate the major department, major salary, department subring, and salary subring, respectively. A significant disadvantage of ring structures is that they can take a long time to search. Updating ring-structured files is normally straightforward. Insertions of new records into the middle of a ring is usually relatively simple. Deletions of records, on the other hand, can be more complex. When a record is deleted from a ring, neither its predecessor record nor the starting record of the ring has to be specified since the predecessor record can be found from any point in the ring by traversing the structure. The deletion performed by traversing a structure and searching for the predecessor record requires that the address of the record to be deleted be saved and compared with the address of the succeeding records accessed. In this case, the entire ring must be


$
b DATA
b DATA
DATA
DATA

Ring structure with a head record and a special pointer in each record

Figure 51. The Ring Structure

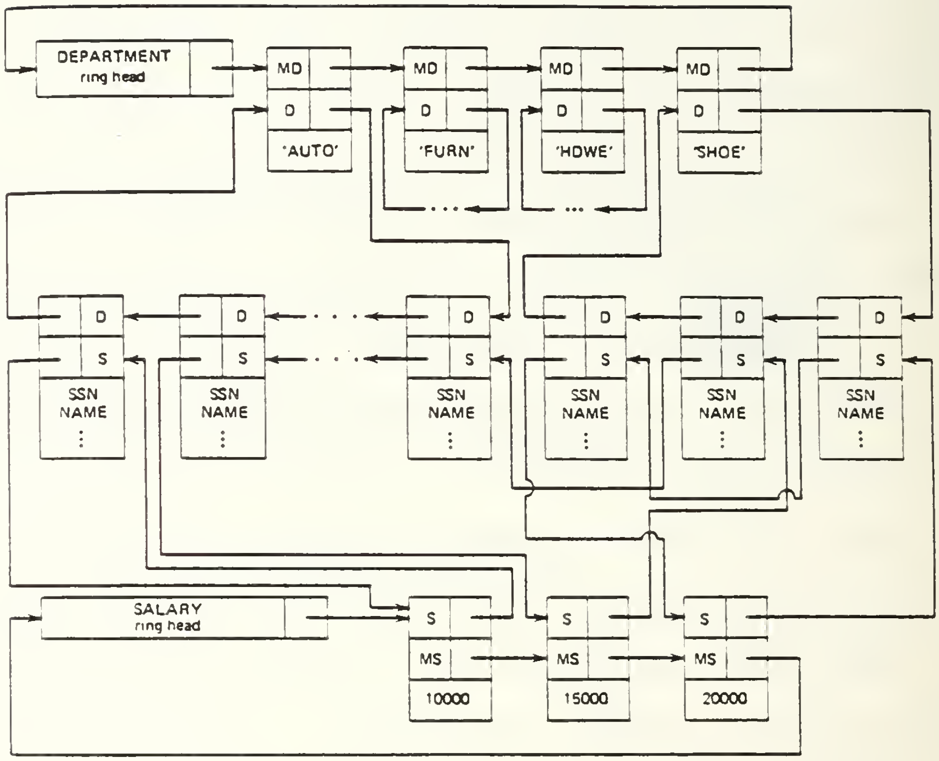

DEPARTMENT
ring head
MD
D
'AUTO'
'FURN'
'HDWE'
'SHOE'
D
S
SSN
NAME
SALARY
ring head
S
MS
10000
15000
20000

Figure 52. The Multiring Structure

traversed for each deletion. Altering the value of data items in a record poses no particular problems.

6. A Summary

The choice of a file organization has a very important effect on the performance and associated costs of a file system. The multilist file organization is satisfactory for systems that do not require extremely fast response times. Nor do they require exact statistics of the records and attributes. Nevertheless, the multilist file organization provides very compact directory despite the volume of data file. On the other hand, the inverted file organization tends to generate large directories. Consequently, fast accesses to the data file are overshadowed by the amount of processing and accesses to the directory. The trade-off is not to "invert" a file down to three level of data items, i.e., field.

The problem of selecting an appropriate file organization depends on the particular users, and their environment. Three very important quantifiable performance measures for selecting a file organization, that should be considered are the total storage costs, the average time to answer a typical query, and the average time to perform an update. The file organizations that have the best access time may require more storage and complicate update, i.e., as access times decrease, storage and update costs go up.

XII. DATA COMPACTION TECHNIQUES

The development of massive information storage and retrieval systems have undergone tremendous growth. Accompanying this growth in the size of the databases has been a large increase in the number of users and duration of usage, resulting in tremendous amounts of data being transferred from computers to terminals. One alternative to this run-away database growth, is to alleviate the data storage problem through the representation of data by more efficient codes, i.e., by **data compression**.

Data compression is a technique of reducing the amount of storage required for a piece of stored data by replacing the data with some representation of the difference between it and the data next to it. Data compression can reduce alphanumeric, numeric and binary data to a shorthand notation. For example, if 30 alphanumeric positions are allocated to the occupational field of a personnel database, for the occupational 9-character description PROFESSOR, there are 21 blank positions. Instead of indicating the occupational title, an equivalent 5-digit data code can be encodes, thereby eliminating 25 character positions.

An example of a numerical and binary compression is as follows: suppose today's date is 1 Jan 1986; numerical representations are 01 01 86, while binary representations are 00001 00001 1010110. Thus, the numeric compression results in 6 numeric characters of storage, while the binary compression results in only 5 bits for the day field (since the day cannot exceed 31), 4 bits for the month field, and 7 bits for the year (permitting a range of 1900-2027).

Accordingly, there are five categories of data compression. These five categories are: *null suppression*, *bit-mapping*, *run-length*, *half-byte packing*, and *pattern substitution*.

1. *The Null-Suppression Technique*

The null-suppression technique has been one of the earliest data compression techniques. As its name implies, it is a technique that scans a data string for repeated blanks or null characters. Upon detection of such a sequence, the null characters are replaced by a special ordered pair of characters. The first is a compression indicator, indicating that null suppression has occurred, and the second indicates the number of null characters encountered. For example, taking the data stream, XYZbbbbbCVF, where b denotes a blank, the compressed data stream would be XYZ@5CVF, where @ represents the special compression indicator character, and 5 the quantity of blanks compressed [Ref. 57].

The technique for decompression is very straightforward. A search is performed for the special character used to denote the null characters. Upon locating the indicator, the next character indicates the number of nulls compressed. Hence, the original string can be reconstructed.

This technique is only effective as long as it is employed for more than two sequentially encountered null characters, since a 2-character compression sequence always results. For nonsequential null characters, a technique known as bit mapping is used.

2. The Bit Mapping

This compression technique is employed when the data consists of a high proportion of specific data types, such as numerics, or a large proportion of a specific character, such as blanks. As its name implies, a bit map is used to indicate the presence or absence of data characters. For example, taking the string DbbKbbHb, where b represents a blank, the bit mapping string would be 10010010^DKH. The zeros represent the location of blanks, and the ^ represents the bit-map character, to denote that bit-mapping compression has taken place. In comparing the two versions of the character string, we note that the original data string of 8 characters of data has been reduced to 4 characters, 3 data characters and the bit-map character [Ref. 57].

To decompress the string the bit map is used to indicate that certain data characters have been encoded upon and must be decoded in order to reconstruct the original data string.

This technique is only effective as long as fixed-size data units are utilized, such as characters, bytes or words. Also, this technique is directly proportional to the percentage of occurrences of a particular character. If there are two or more significant occurrences of other characters, only the character with the highest occurrence can be compressed. Another compression technique called run-length encoding, can handle adjacent redundancy of occurrences of all characters in a data stream [Ref. 58].

3. The Run-Length Encoding Technique

The run-length encoding technique is a data compression method that reduces any type of repeating character sequence. The method employed is similar to the null suppression, in that it uses a special character to denote that this type of compression has occurred. The compression indicator is followed by one of the repeating characters which has been in the encountered string of repetitious characters. Finally, a count character signifies the number of times the repeated character occurred in the sequence. The general character format is:

a special indicator, any repeating data character, and the character count. For each of these codes, the numerous unassigned characters with unique bit representations can be used [Ref 59].

4. The Half-Byte Packing

This compression technique is used when a portion of the bit pattern that represents certain characters in a character set becomes repetitive. It is actually a derivative of the bit-mapping method. For example, considering taking the EBCDIC (Extended Binary-Coded Decimal Interchange Code) character set, which is an IBM scheme for representing characters by combinations of bits. The half-byte packing can be utilized, since the first four bit positions are all set to binary ones to represent numerics.

To compress data into half bytes, up to 15 sequential numeric or predefined data characters in a string can be compressed. The reason of 15 characters results from the use of a 4-bit, half-byte counter to denote the number of characters being compressed. The general format is as follows: special character indicating half-byte compression, half-byte counter, up to 255 numerics packed. For example, taking the numeric 2112860, the binary is, 11110010 11110001 11110001 11110010 11111000 11110110 11110000, and the compressed data string is S 0111 0010 0001 0001 0010 1000 0110 0000, where S is the special character and 0111 is the number of packed numerics (7). The number of bits has been reduced from 56 bits to 40 bits.

The half-byte packing can also be used when data characters do not have a repetitive bit structure, such as the ASCII (American Standard Code for Information Interchange) tables, which is a standard scheme to represent characters by combinations of bits. ASCII tables employ 7 bits. A method of doing this is to predefine the occurrence of the dollar sign, asterisk, comma, the decimal point and the 10 numerics. For example, given the amount $1,234.56, the ASCII code would be 0100100 0110001 0101100 0110010 0110011 0110100 0101110 0110101 0110110, and the compressed data string is, 00100 0001 01100 0010 0011 0100 01110 0101 0110, where three 5-bit stream represent a dollar sign, a comma and a decimal point, respectively, and 4-bit streams represent the respective numerics [Ref. 58].

5. The Pattern Substitution

This compression technique substitutes a special code for a predefined character pattern, that is, common key words or phrases can be replaced by a special code. To use this technique, a pattern table is required, which contains a set of list arguments (words or phrases to be compressed) and a set of function values (special character codes). For example, given a limited pattern table with list arguments: at, all, and, both; and function values S1, S2, S3, S4, respectively, the data stream, all naval officers, both male and female, at NPGS, becomes: S2 naval officers S4 male S3 female S1 NPGS. The employment of pattern substitution can be highly advantages when texts with known repeating patterns are stored in the database [Ref. 57].

6. The Summary

When the data compression is used to reduce storage requirements, the overall processing time is also reduced. This is because the reduction in storage results in a reduction of disk access attempts. Although, the compression techniques result in additional program instructions being executed, it is significantly less than the time required to access and transfer data. Hence, a reduction of storage requirements in the case also results in a reduction of processing times. The most effective means of employing these compression techniques is to combine them as they are needed, depending on the data to be stored.

XIII. DATA MODELS FOR DATABASES

A **data model** is an abstract representation or description of a database that describes how the data is put together. The purpose of the data model is first to accurately and completely represent required data of a database and second to allow the database to be understandable. The data model also dictates the design of the corresponding manipulation language (DML), since each DML operation is defined in terms of its effect on those modeled data. A DML is a language used to access and to update a database. A DML may be procedural or nonprocedural. To manipulate a database using a procedural DML, a user normally writes short segments of DML statements that traverse the modeled database in order to locate the record to be retrieved or updated. Nonprocedural DMLs are easier for a user to use in manipulating a database. With this type of DML, the user does no have to traverse a database; instead, the user specifies only what is wanted and allows the system to decide how to obtain it.

The more procedural a DML, the simpler it is to implement since the user directs the database traversal, step by step, on how to obtain data. A nonprocedural DML is more complicated to implement since it places the responsibility of determining how to obtain the data and, therefore, how to optimize the search, on the database system.

Database systems are categorized according to the approach they adopt in data model and accompanying DMLs. The three most reknowned approaches are, **the hierarchial data model**, which usually supports the procedural DML, because, in general, it is too inefficient to perform database accesses in a strictly nonprocedural manner, **the network data model**, which also supports the procedural DML, because of its efficiency, and **relational data model**, which supports, the nonprocedural DML because searching, of tables (relations) does not require traversals and is easily expressed in a nonprocedural manner.

A. HIERARCHIAL DATA MODEL

The hierarchial data model is a tree structure organization which represents the data as a set of nested one-to-many (1:M) and one-to-one (1:1) relationships. A one-to-many association from a record of type A to a set record of type B means that at each period in time, a given record of A is associated with zero, one, or a number of records of B; This association is represented with a double-headed arrow. A one-to-one association, on the other hand, means that for a specified period of time, a given record of A is associated with one and only one record of B.

This association is represented with a single-headed arrow. In implementations, associations are carried out by record-address pointers. We term record type B directly below record type A.

Hierarchies, although they are a familiar structures, are very explicit in a data model. If one record type is not directly below any record type in the hierarchy, then no accesses to the record type is possible. Multiple (subtrees) subhierarchies are allowed, but there can only be one parent, that is one root - the apex of the hierarchy. Figure 53 illustrates the above points.

The basic operation on a hierarchial database is a tree walk (traversal). The search starts at the root and continues to all its descendants of the given record type, until the query is satisfied. This model uses extensive pointers. These pointers could point to a dependent child record, to the next record, or to the parent record. These links (pointers) in a hierarchial structure are unidirectional from parent to child (descendant). This convention causes certain relationship to be hard to extract from the database, although they may be implied in the data. This anomaly affects each of the basic storage operations, insert, delete, and update.

Insertions are not possible without introducing a special dummy customer to insert data concerning a new order, until the order supplies some customer (see Figure 53). Deletions are corollaries to insertions. The only way to delete a shipment is to delete an order, and if the only order is deleted, all information dependent on that order is also lost. Updating a specific record presents the problem of either searching the entire model to find every occurrence of that change or introducing an inconsistency. For example, to change the city for a supplier/vendor to make deliveries for the orders, either the entire database is searched for that supplier or that supplier may be shown in one city at one point and in another city at another point.

Normally, hierarchial databases use the inverted file technique for indexing as a way to avoid prolong traversals. In Figure 53, the leaf (descendant) product can be indexed by product#, thereby allowing any record to find its parent [Ref. 60].

B. NETWORK DATA MODEL

The network data model represents data as a set of record types and pairwise relationships between records of two record types. It is a more general structure than a hierarchy because a given record occurrence may have any number of immediate parents, as well as any number of immediate dependents. This model is not limited to just one parent. Hence, it can have many-to-many relationships, as in Figure 54, which is the network version of the hierarchial model as described in Figure 53. The network model also supports one-to-many relationships.

Although the network data model does not have the anomalies as in the hierarchial model, storage operations are not as straightforward as expected. Insertions are simple. To Insert data concerning a new supplier, a new supplier record occurrence is created. Deletions, on the other hand, confront the user with the choice strategy, that is, to delete shipment associating product with vendor, the problem is that there are two strategies for locating this occurrence, one that starts at the supplier and scans its chain looking for a pairwise relationship to the product, and one that starts at the product and scans its chain looking for a pairwise relationship to the supplier. The choice can be significant. Updating is straightforward.

Retrieval with most database systems of network databases begin with accessing a parent record via some entry point into the database. Then the search continues through the relevant database records by getting the first or next record in relationships. Due to its complexity, keeping track of where in the database that the current search is taking place, is a chore [Ref. 61].

The purpose of this model is to convey what is implemented in the database. Many relationship types can be

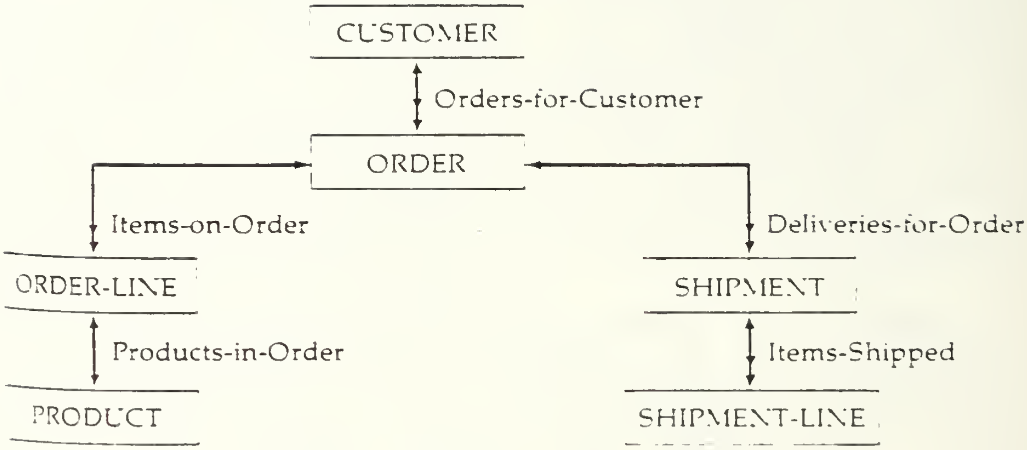

CUSTOMER
Orders-for-Customer
ORDER
Items-on-Order
Deliveries-for-Order
ORDER-LINE
SHIPMENT
Products-in-Order
Items-Shipped
PRODUCT
SHIPMENT-LINE

Figure 53. The Hierarchial Data Model

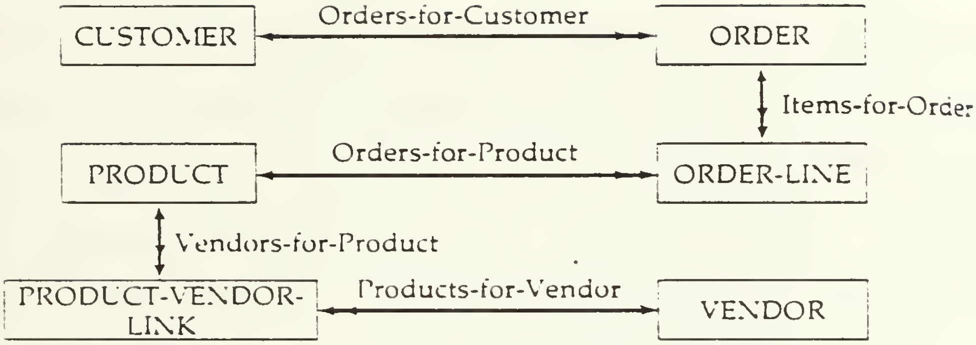

CUSTOMER
Orders-for-Customer
ORDER
Items-for-Order
PRODUCT
Orders-for-Product
ORDER-LINE
Vendors-for-Product
PRODUCT-VENDOR-LINK
Products-for-Vendor
VENDOR

Figure 54. The Network Data Model

easily depicted, and both relationship type and the record type are explicitly stated.

C. THE RELATIONAL DATA MODEL

A relational data model, as its name implies, uses the concept of *relations* to represent files. A relation is a two-dimensional table, which contains rows (tuples) that correspond to records of a flat file. A flat file contains no repeating groups, i.e., there is exactly one value at every row and column (attribute) position and never a set of values. A table represents one record type and each row represents a particular record of that type. Columns are attributes, with all values in a column having the same domain, which is the set of possible values for an attribute. An important feature of a relational database is that associations between rows are represented solely by data values in columns drawn from a common domain. Figure 55 illustrates the relational version of the hierarchial database, as described in Figure 53. In this example, CUSTOMER, PRODUCT, and VENDOR are basic relations that exist independently from all other data. The ORDER relation, can also exist independently, but for one of its attributes, CUSTOMER#, for which no more than one tuple may have the same value. This attribute implements the Orders-for-Customer relationship in Figure 53, i.e., any value of CUSTOMER# found in an ORDER tuple logically should exist as a CUSTOMER# in some unique existing CUSTOMER tuple. The other relations work in a similar manner [Ref. 62].

The files in a relational database may be organized in any of the known file organization techniques, such as heap, sequential, index sequential, hash etc. As for storage operations, insertions, deletions, and updates are all easily handled.

D. A SUMMARY

In the hierarchical and network approaches certain relationships are represented by means of links. Basically such links are capable of representing one-to-many associations; the difference between the two approaches is that with the latter, links may be combined to model more complex many-to-many associations, whereas this is not possible with the latter. Another difference, not emphasized is that links are generally named in a network and anonymous in a hierarchical.

The relational model organizes data into tables of like data and supports intertable linkages through common data occurrences rather than pointers.

In short, the hierarcnial data model is the most natural, the most familiar and best understood one when it is used to represent the engineering design databases. In such a database, we have records on assemblies, records on

CUSTOMER(CUSTOMER#,CUSTOMER-ADDRESS,
CUSTOMER-DETAILS)

ORDER(ORDER#,CUSTOMER#,ORDER-DATE,
DELIVERY-DATE,TOTAL-AMOUNT)

PRODUCT(PRODUCT#,DESCRIPTION,PRICE,
QUANTITY-ON-HAND)

ORDER-LINE(ORDER#,PRODUCT#,
QUANTITY-ORDERED,EXTENDED-PRICE)

VENDOR(VENDOR#,VENDOR-NAME,VENDOR-CITY)

SUPPLIES(VENDOR#,PRODUCT#)

Figure 55. The Relational Data Model

subassemblies, records on components of a subassembly, records on parts of a component and records on individual parts. These records naturally form a hierarchy of engineering design database. A hierarchial database system can best be used to access, manipulate and update the engineering design databases. The network model is the most natural, offers flexibilility for an inventory control application. In inventory control a product may have many suppliers and a supplier may produce many products. A network database system can easily manage, access and update these many-to-many relationships. The relational data model is efficient, understandable, for interactive use and ad hoc queries. It is a relative new entry to database management. The current commercial database machines are relational, for example, the Britton-Lee Intelligent Database Machine and the Teradata database computer, DBC 1012.

XIV. DIFFERENTIAL FILES

A failure or break-down of a database can be catastrophic, if there is not any kind of recovery techniques to recover the data that has been broken down. A recovery technique can be used to restore data, in such a situation, to a usable state. This is accomplish by maintaining recovery data to make recovery possible. It provides recovery from a failure which does not affect the recovery data or the mechanisms used to maintain the recovery data and to restore the states of the data in the database. The most popular recovery technique is the use of differential files.

A differential file consists of a relatively small storage area, in which all database alterations are recorded. It is an efficient method for storing a large and changing database. It is analogous to an errata list for a book. Rather than print a new edition each time that a change in text is desired, a publisher distributes an errata sheet along with the book that identifies corrections in the book by page and line number.

Under a differential database representation, the main files are kept unchanged until reorganization, which can occur basically at any time, from hours to months, dependent on usage. All changes that would normally be made to a main file as a result of a transaction are instead registered in a differential file. The differential file is always searched first when data is to be retrieved. Data not found in the differential file is retrieved from the main database. Verhofstad in [Ref. 64], describes an efficient hashing method to implement the recovery technique.

A. THE CONSTRUCTION OF A DIFFERENTIAL FILE

To implement a differential file, a small associative memory in the form of a bit map accessed by a hashing scheme is used. To reduce the probability of making an unnecessary search in the differential file, the database system checks the bit map to see whether the bits for a record are set or not before accessing that record (see Figure 56). If the bits are set the record is probably in the differential file; otherwise, the main file would have to be searched. The hashing function maps the record address onto a number of bits in the bit map.

B. ADVANTAGES OF A DIFFERENTIAL FILE

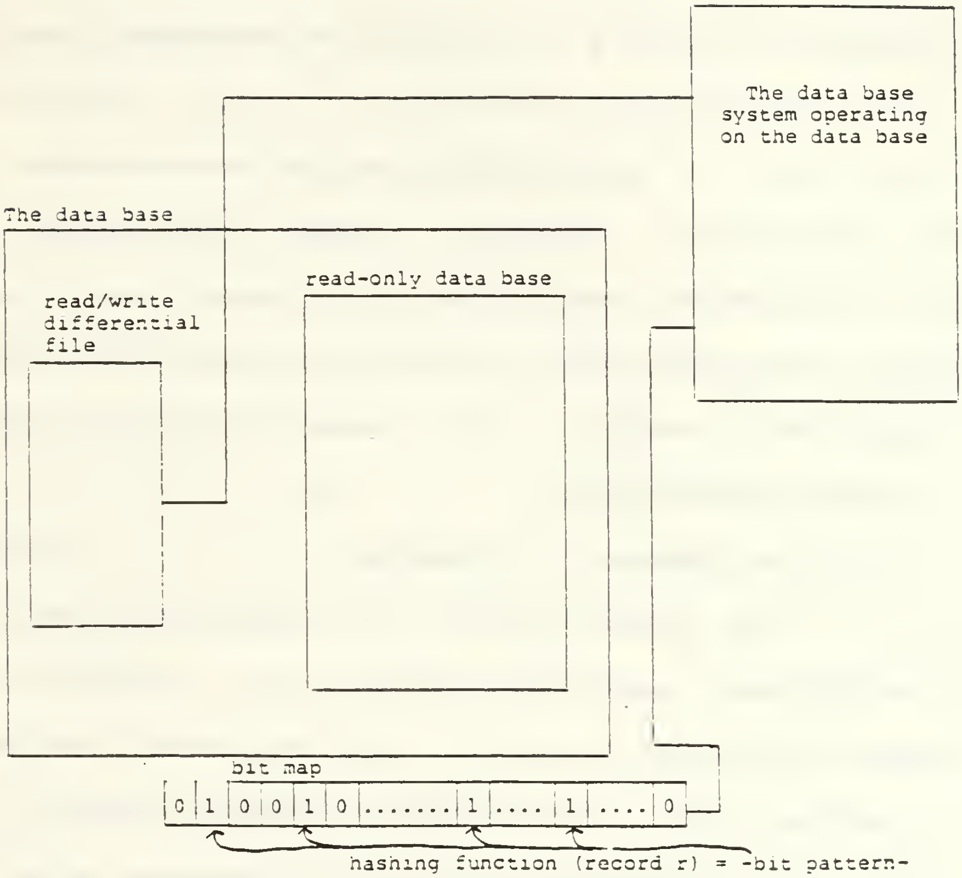
Severance and Lohman in [Ref. 65], describe five advantages of differential files. The first three advantages relate to the database integrity, i.e., the correctness of data to be recovered. They are, reduction of backup costs, speedup of recovery, and minimization of serious data loss. The other two advantages are operational; a differential file can provide increased data availability and simultaneously reduce storage and retrieval costs.

1. The Reduction of Backup Costs

To recover data from a database that has failed, the status of a previous state must be reloaded. The method available to generate previous states employs either a total dump, or an incremental dump.

A total dump of the database takes place when the backup copy of the entire database is reloaded. The frequency with which the database is copied to its backup file is dependent upon the database usage. When it is impractical to dump the entire database, an incremental dumping is performed, in which sequential sections of changes made to a database are periodically dumped. Frequent dumps permit fast recovery, but are associated with a higher system overhead.

A differential file can drastically reduce the cost to backup a large database, since the time required for a dump is proportional to the volume of data being copied. This is particularly true when the proportion of records changed during a backup period is small. For example, a total dump may require up to 6 hours, assuming that updates are made 5 days per week, 10 hours per day. A differential file on the other hand, for the same period of time, could be dumped in less than two minutes. Moreover, a


The data base system operating on the data base
The data base
read-only data base
read/write differential file
bit map
0 1 0 0 1 0 .... 1 .... 1 0
hashing function (record r) = -bit pattern-

The bit map suggests that record r is in the differential file, because the bits set in the bit patterns produced by the hashing function are set in the bit map.

A differential file technique using a hashing scheme.

Figure 56. The Differential File

differential file would occupy less than one disk as compared to over 50 for a total dump.

A differential file also permits both realtime dumping and reorganization with concurrent updates. During a conventional backup procedure, no updating is possible. But by building a "differential-differential file", updating can continue. For most applications, this file will be quite small and can be reasonably stored in the main memory. Acting as a cache store during the dump, it is scanned before every retrieval. When the dump has been completed, its records is incorporated into the main differential file. The same procedure would apply for online organization.

2. The Speedup of Recovery

The major portion of recovery time for for a traditional recovery method is spent individually reapplying updates to a small fraction of the restored records. This small subset of changed records guarantees that even localized physical damage will require a lengthy recovery procedure. A differential file, on the other hand, by concentrating updates in a small physical area minimizes the critical exposure area of the database. Most physical damage can be quickly repaired with a localized backup-copy procedure. Also, the critical area can be allocated to a more reliable device type than is practical for the larger main file, and this critical area can be duplexed to provide the most valuable redundancy for a marginal increase in operating costs. Moreover, since the use of a differential file can dramatically reduce the cost of dumping a large database, an inexpensive dump procedure can be invoked frequently to reduce the number of changes to be remade in the event of a database loss.

3. An Increase of Data Availability

Traditional online updating requires complex software to assure data recoverability. Therefore, updates are normally batched for end-of-day processing, to minimize overhead. With a differential file, since the main file and its associative index is not affected by updates, a less complex and more efficient software procedure is required, thereby enhancing the achievement of a greater density of data storage. Neither free space nor record pointers need to be allocated for record growth. Moreover, the cost reduction that a differential file provides will greatly enlarge the realm of database systems.

4. The Summary

A differential file is the most popular representation of database recovery techniques. By localizing modifications in a small storage area and physically isolating it from the main file, it is possible to realize some important benefits as aforementioned. A differential file is conceptually simple, i.e., a dominant characteristic of a successful implementation.

XV. THE SUMMARY OF THE THESIS

Computerized database applications have grown over the past thirty years to a point where they have now become a pervasive influence in our society. For the past thirty years the conventional magnetic recording has, almost exclusively, fulfilled the online storage requirements of this database applications community.

As the range of applications has grown, a continuing concern has been the cost and access time of the online database storage. A wide range of technologies have been investigated to address this challenge. As rapid as the progress in the storage technology has been, the need for more capacity with faster access has increased even greater.

While the conventional magnetic recording is entering yet another phase of explosive growth in applications and advances in technology in order to meet these stringent requirements, the optical disks have begun to challenge the magnetic media. There are pressures to break free of the limitations of magnetic storage where large volumes of data are involved. These pressures come from the continuing growth of conventional storage, existing requirements of large corporate and governmental databases, and the development of new applications such as storage of digitized documents where large volumes of data must be stored at a lower cost. Such applications often demand a cost, capacity and performance combination that is difficult to achieve magnetically. The optical storage is able to provide a performance that is competitive with the performance of magnetic recording. In fact, emerging optical technologies are already capable of replacing magnetic disks in certain applications. However, there is no single technology that is right for all applications. Whereas technologies such as RAMs are fast and technologies such as magnetic disks and optical disks are inexpensive, we know of nothing that is both fast and inexpensive. Thus, database installations often have available a wide range of different storage technologies. The needs of an application must be analyzed to determine the appropriate technology to use.

In this thesis we have examined high-volume, on-line storage media of current and emerging technologies and software techniques for supporting these on-line, high-capacity storage and access requirements. In the first part, we have analyzed such media as vertical magnetic recording, thin film media, optical data disks, magneto-optic disks, bubble and Bernoulli-effect disks. Then, comparisons and evaluations of products and product categories have been illustrated. In the second part, we haved reviewed the modern software techniques for on-line database storage and access. We haved explored the techniques for data abstraction, data access and retrieval, data compaction, data models for storage, and differential files. There are advantages and disadvantages to all technologies and techniques. The individual application of the organization must be used to dictate the specific requirements, along with its financial constraints. With these requirements, the organization can then take the advantage of certain strong points of hardware technologies and the software techniques and utilize them in meeting the requirements. This thesis has provided a comprehensive and up-to-date analysis of the strong points and weak points of the hardware technologies and software techniques which in turn make it easier for an organization in meeting its requirements, for online storage and access.

LIST OF REFERENCES

1. White, R., "Disk-Storage Technology", Scientific American, Vol. 243, No. 2, pp. 138-148, August 1980.

2. Digest of Papers, Intellectual Leverage - The Driving Technologies, pp. 512-517, Computer Society Press, 1984.

3. Mallinson, J. C., "The Next Decade in Magnetic Recording", IEEE, pp. 89-92, 1984.

4. Digest of Papers, "Topical Meeting on Optical Data Storage", pp. MA2-1 to MA2-4, Optical Society of America, 1983.

5. Digest of Papers, Mass Storage Systems, National Center for Atmospheric Research, Boulder, Colorado, 1980.

6. Digest of Papers, Mass Storage Systems, Computer Socoety Press, pp. 312-376, 1977.

7. Airel, D. L., "Much Maligned Bubble Memories Find a Home", Data Communications, v.12, pp 139-141, Jan 1983.

8. Dekker, P., George, P. K. and Middelhoek, S., Physics of Computer Memory Device, pp.157-189. Academic Press, 1976.

9. Intel Corporation, Solutions, pp. 20-23, Order No. 231354-005, Sept/Oct 1985.

10. Scott, R., "All About Bubble Memory Devices: Part I & II", *Radio-Electronics*, pp. 39-42, pp 75-78, Oct/Nov 1982.

11. Eschenfelder, A. H., *Magnetic Bubble Technology*, ISBN 3-540-09822-4, pp. 163-224, 1980.

12. Susuki, T., "Perpendicular Magnetic Recording", *IEEE Transactions on Magnetics*, pp. 675-680, v. Mag20, no. 5, Sept 1984.

13. Iwasaki, S., Nakamura, Y.; *IEEE Transactions on Magnetics*, v. Mag-13, no. 5, pp. 1272-1277, 1977.

14. *McGraw-Hill Encyclopedia of Electronics and Computer*, pp.451-456, McGraw-Hill Book Company, 1984.

15. "Focus on Optical Storage", *Creative Computing*, v. 11, no. 9, pp. 43-49, September 1985.

16. Digest of Papers, *Intellectual Leverage - The Driving Technologies*, pp. 508-511, Computer Society Press, 1984.

17. Hoagland, A., Lecture Notes, "Magnetic Recording-The Future", Institute For Information Storage Tecnnology, pp. 8-1 to 8-85, March 1985.

18. Walter, G., "Optical Digital Storage of Office and Engineering Documents", *Journal of Information and Image Management*, v. 17, no. 7, pp. 27-35, April 1984.

19. *Computer Shopper*, "Is There a WORM in your Future", pp. 20-21, August 1985.

20. Bell, A., Lecture Notes, "Optical Recording and Storage Media", Institute For Information Storage Technology, pp. 7-1 to 7-128, March 1985.

21. Chen, M., Marrello, V., and Gerber, U., *Appl. Phys. Letter*, no. 41, pp. 892-896, 1982.

22. Bell, A., "Critical Issues In High-Density Magnetic and Optical Storage", *Laser Focus/Electro-Optics*, v. 12, no. 9, pp. 125-136, Sept. 1983.

23. Digest of Papers, "Technological Leverage-A Competitive Necessity", *Compcon 85*, pp. 32-38, Feb 1985.

24. Hartman, M., "Erasable Magneti-Optical Recording Media", *IEEE Transactions on Magnetics*, v. MAG-20, no. 5, pp. 1013-1018, Sept. 1984.

25. Imamura, N., *Proceedings of the International Conference on Ferrites*, Kyoto, Japan, pp. 796-780, Sept/Oct 1980.

26. Giglio, L., "Product Focus", *Information Week*, pp. 42-45, Sept 2, 1985.

27. Cormier, D., "High-Speed Digital ICs", *Electronic Technology For Engineers And Engineers Managers (EDN)*, pp. 111-132, Aug 22, 1985.

28. Moore, S., "The Mass Storage Squeeze", *Datamation*, pp.68-79, Oct. 1984.

29. Bursky, D., "Memory Technology-Nonvolatile Memories", *Electronic Design*, pp. 123-144, Aug 23, 1984.

30. *Infoworld*, "The Bernoulli Box from Iomega", v. 5, no. 41, Oct. 10 1983.

31. Edmunson, H., "New Abstracting Methods", *Machinery*, v. 16, no. 2, pp. 265-285, 1969.

32. Rowe, N., "Antisampling for Estimation", *NPGS*, Oct. 1984.

33. Whang, K., and Sagalowicz, D., "A Close Noniteratve Formula to Estimate Block Accesses", *Communications of the ACM*, v. 26, no. 11, pp. 940-944, Nov. 1983.

34. Rowe, N., "Rule-Based Statistical Calculations on a Database Abstract", June 1983, Report no. STAN-CS-83-975.

35. Harris, R., "A Method for Managing Narrative Information", Analytics Inc., Contract no. N00014-71-C-0186, Sept. 1972.

36. *Journal of the American Society for Information Science*, "Automatic Abstracting System", v. 22, no. 4, pp. 260-274, Jul/Aug 1971.

37. Zadeh, I.A., "Fuzzy Sets", *Information and Control*, 1968.

38. Pollock, J. and Zamora, A., "Automatic Abstracting Research", 1975.

39. Maeda, T., "An Automatic Method For Extracting Significant Phrases", 1983.

40. Reeker, L., Zamora, E. and Blower, P., "SIE", *Proc. Conf. Applied Natural Language Processing*, pp. 108-116, Feb 1-3, 1983.

41. Claybrook, B.G., *File Management Techniques*, John Wiley and Sons, 1983.

42. Deitel, H. M., *Introduction to Operating Systems*, Addison-Wesley Publishing Co., 1984.

43. McFadden, F. and Hoffer, J. A., "Database Management", 1985.

44. Larson, P.,"Linear Virtual Hashing", 1979.

45. Larson, Per-Ake, "Dynamic Hashing", BIT, Feb 1978.

46. Larson, P. A., "Analysis of Linear Hashing With Partial Expansion", Sept 1980.

47. Burkhard, W., "Interpolation Based Hashing", Dept of Electrical Engineering, University of Cal, Feb 1983.

48. Fagen, R., "Extendible Hashing", *ACM Transactions on Database Systems*, V. 4, No. 3, Sept 1979, pp.315-344.

49. Mendelson, H., "Analysis of Extendible Hashing", IEEE Transactions On Software Engineering, Vol. Se-8, No. 6, Nov. 1982.

50. Chen, W. and Vitter, J., "Analysis of Coalesced Hashing", *ACM Transactions on Database Systems*, V. 9, No. 4 Dec 1984.

51. Date, C. J., *An Introduction to Database Systems*, Addison-Wesley Publishing Co., 3rd edition, 1982.

52. Ullman, J., *Priciples of Database Systems*, Computer Science Press, Standford University, 1982.

53. Wetherbe, J., *System Analysis and Design*, West Publishing Co., 1984.

54. Claybrook, B., *File Management Techniques*, John Wiley and Sons, pp. 131- 187, 1983.

55. Koymen, K., "Ksam: A B+ Tree- Based Keyed Sequential-Access Method", *AFIPS Conference Proceedings*, July 9 - 12, 1984.

56. Deogun, J., Raghavan, V., and Tsou T., "Organization of Clustered Files", *ACM Transactions on Database Systems*, v.9, no. 4, pp. 646-672,Dec 1984.

57. Aronson, J., "Data Compression - A Comparison of Methods", *National Bureau of Standards*, pp. 269-296, June 1977.

58. Held, G., *Data Compression - Techniques and Applications*, A Wiley Heyden Publication, 1983.

59. Rubin, F., "Experiments in Text File Compression", *Communications of the ACM*, v. 19, no. 11, pp.617-623, 1976.

60. Hoffer, J. and McFadden, F., *Database Management*, The Benjamin/Cummings Publishing Co., 1985.

61. Ullman, J., *Database Systems*, Computer Science Press, Second Edition, 1982.

62. Date, C., "An Introduction to Database Systems", Addison-Westley Publishing Co., Third Edition, 1981.

63. Tsichritzis, D. and Lochovsky, F., *Data Models*, Prentice Hall, Inc. 1982.

64. Verhofstad, J., "Recovery Techniques for Database Systems", *ACM Computing Surveys*, v. 10, no. 2, 1978.

65. Severance D. and Lohman G., "Differential Files", *ACM Transactions on Database Systems*, v. 1, no. 3, Sept. 1976.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5100	2
3.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93943-5100	2
4.	Curriculur Officer, Code 37 Computer Technology Naval Postgraduate School Monterey, California 93943-5100	1
5.	Professor David K. Hsiao, Code 52Hq Computer Science Department Naval Postgraduate School Monterey, California 93943-5100	2
6.	Steven A. Demurjian, Code 52 Computer Science Department Naval Postgraduate School Monterey, California 93943-5100	2
7.	Christopher V. Feudo 6305 Moon Road Columbus, Ga. 31909	5

602-197

216347

Thesis
F2697
c.1

Feudo
Modern hardware technologies and software techniques for online database storage and access.

216347

Thesis
F2697
c.1

Feudo
Modern hardware technologies and software techniques for online database storage and access.

thesF2697
Modern hardware technologies and softwar
3 2768 000 64825 7
DUDLEY KNOX LIBRARY